中标麒麟高级服务器操作系统软件 V7.0

系统管理员手册

中标软件有限公司

2020 年 5 月

目录

中标麒麟最终用户使用许可协议

尊敬的中标麒麟高级服务器操作系统用户：

首先感谢您选用由中标软件有限公司开发并制作发行的中标麒麟高级服务器操作系统产品。

请在打开本软件介质包之前，仔细阅读本协议条款以及所提供的所有补充许可条款（统称“协议”）。一旦您打开本软件介质包，即表明您已接受本协议的条款，本协议将立即生效，对您和本公司双方具有法律约束力。

1.使用许可

按照已为之支付费用的用户数目及计算机硬件类型，中标软件有限公司（下称“中标软件”）向您授予非排他、不可转让的许可，仅允许内部使用由中标软件提供的随附软件和文档以及任何错误纠正（统称“本软件”）。

— 软件使用许可

在遵守本协议的条款和条件的情况下，中标软件给予贵机构非独占、不可转让、有限的许可。

— 教育机构使用许可

在遵守本协议的条款和条件的情况下，如果贵机构是教育机构，中标软件给予贵机构非独占、不可转让的许可，允许贵机构仅在内部使用随附的未经修改的二进制格式的软件。此处的“在内部使用”是指由在贵机构入学的学生、贵机构教员和员工使用软件。

— 字型软件使用

软件中包含生成字体样式的软件（“字型软件”）。贵机构不可从软件中分离字型软件。贵机构不可改动字型软件，以新增此等字型软件被作为软件的一部分交付予贵机构时所不具备的任何功能。贵机构不可将字型软件嵌入作为商业产品提供以换取收费或其他报酬文件。

2.限制

本软件受到版权（著作权）法、商标法和其他法律及国际知识产权公约的保护。中标软件和/或其许可方保留对本软件的所有权及所有相关的知识产权。对

于中标软件或其许可方的任何商标、服务标记、标识或商号的任何权利、所有权或利益，本协议均不作任何授权。

3.关于复制、修改及分发

如果在所有复制品中维持本协议书不变，您可以且必须根据《GNU GPL-GNU 通用公共许可证》复制、修改及分发中标麒麟高级服务器操作系统产品中遵守《GNU GPL-GNU 通用公共许可证》协议的软件，其他不遵守《GNU GPL-GNU 通用公共许可证》协议的中标麒麟高级服务器操作系统产品必须根据符合相关法律之其他许可协议进行复制、修改及分发，但任何以中标麒麟高级服务器操作系统产品为基础的衍生发行版未经中标软件有限公司的书面授权不能使用任何中标软件有限公司的商标或其他任何标志。

特别注意：该复制、修改及分发不包括本产品中包含的任何不适用《GNU GPL-GNU 通用公共许可证》的软件，如中标麒麟高级服务器操作系统产品中包含的输入法软件、字库软件、第三方应用软件等。除非适用法律禁止实施，否则您不得对上述软件进行复制、修改（包括反编译或反向工程）、分发。

4.有限担保

中标软件向您担保，自购买之日起九十（90）天内（以收据副本为凭证），本软件的存储介质（如果有的话）在正常使用的情况下无材料和工艺方面的缺陷。除上述内容外，本软件按“原样”提供。在本有限担保项下，您的所有补偿及中标软件的全部责任为由中标软件选择更换本软件介质或退还本软件的购买费用。

5.担保的免责声明

除非在本协议中有明确规定，否则对于任何明示或默示条件、陈述及担保，包括对适销性、对特定用途的适用性或非侵权性的任何默示担保，均不予负责，但上述免责声明被认定为法律上无效情况除外。

6.责任限制

在法律允许范围内，无论在何种情况下，无论采用何种有关责任的理论，无论因何种方式导致，对于因使用或无法使用本软件引起或与之相关任何收益损失、利润或数据损失，或者对于特殊的、间接的、后果性的、偶发的或惩罚性的损害

赔偿，中标软件或其许可方均不承担任何责任（即使中标软件已被告知可能出现上述损害赔偿）。根据本协议，在任何情况下，无论是在合同、侵权行为（包括过失）方面，还是在其他方面，中标软件对您的责任将不超过您就本软件所支付的金额。即使上述担保未能达到其基本目的，上文所述限制仍然适用。

7.终止

本协议在终止之前有效。您可以随时终止本协议，但必须销毁本软件的全部正本和副本。如果您未遵守本协议的任何规定，则本协议将不经中标软件发出通知立即终止。终止时，您必须销毁本软件的全部正本和副本。

8.管辖法律

与本协议相关的任何诉讼均受适用的中华人民共和国法律管辖。任何其它国家和地区的选择法律的规则不予适用。

9.可分割性

如果本协议中有任何规定被认定为无法执行，则删除相应规定，本协议仍然有效，除非删除防碍各方愿望的实现（在这种情况下，本协议将立即终止）。

10.完整性

本协议是您与中标软件就其标的达成的完整协议。它取代此前或同期的所有口头或书面往来信息、建议、陈述和担保。在本协议期间，有关报价、订单、回执或各方之间就本协议标的进行的其他往来通信中的任何冲突条款或附加条款，均以本协议为准。对本协议的任何修改均无约束力，除非通过书面进行修改并由每一方的授权代表签字。

11.商标和标识

贵机构承认并与中标软件有以下共识，即中标软件拥有中标软件、中标麒麟商标，以及所有与中标软件、中标麒麟相关的商标、服务标记、标识及其他品牌标识(“中标软件标记”)。贵机构对中标软件标记任何使用都应有利于中标软件。

12.源代码

本软件可能包含源代码，其提供唯一目的是在符合本协议条款之规定时供参

考之用。源代码不可再分发，除非在本协议中有明确规定。

13.因侵权而终止

如果本软件成为或在任一方看来可能成为任何知识产权侵权索赔之标的，则任一方可立即终止本协议。

14.Java 技术限制

贵机构不可更改 Java 平台界面（简称 JPI，即指明为 java 包或 java 包的任何子包中的类），无论通过在 JPI 中创建额外的类，还是通过其他方式导致对 JPI 中的类进行增添或更动，均为不可。如果贵机构创建一个额外的类以及一个或多个相关的 API，而它们扩展 Java 平台的功能；并且可供第三方软件开发者用于开发可调用上述额外 API 的额外软件，则贵机构必须迅即广泛公布对此种 API 的准确说明以供所有开发者免费使用。贵机构不可创建或授权贵机构的被许可人创建以任何方式标示为 java、javax、sun 的额外的类、界面、子包或 Sun 在任何命名约定中指明的类似约定。参见 Java 运行时环境二进制代码许可的当版本（目前位于 http://www.java.sun.com/jdk/index.html），以了解可与 Java 小程序和应用程序共同分发运行时代码的可供情况。

特别提示说明

本手册主要面向系统管理员及相关技术人员，如本手册未能详细描述之处，有需要请致电中标软件有限公司技术服务部门。

重要：

本手册中命令、操作步骤等举例仅供参考，命令执行的输出信息等在不同 CPU 平台或因操作系统或组件的版本升级可能有少许差异；本手册尽量加以说明。如有差异之处，请以中标麒麟高级服务器操作系统软件 V7.0 在具体 CPU 平台上实际操作或输出信息为准。

第一章 基本系统配置

这部分涵盖了基本的系统管理任务，如键盘配置、日期和时间配置、用户和组群配置以及授权配置。

1.1. 系统地区和键盘配置

系统地区配置是指系统服务和用户界面的语言环境配置。键盘布局配置是指文本控制台和图形用户界面的键盘布局规则。这些设置可以通过修改/etc/l ocale.conf 配置文件或使用 localectl 命令。此外，您可以在用户图形界面来执行任务，详情请参考安装手册。

1.1.1. 配置系统地区

系统地区配置文件为/etc/locale.conf，在系统启动时引导 systemd 守护进程。这个配置文件可以被每一个服务或者用户继承，单个服务或者用户也可修改配置文件。例如语言为英语，地区为德国的/etc/locale 文件的配置内容如下：

```
LANG=de_DE. UTF- 8
LC_MESSAGES=C
```

LC_MESSAGES 选项决定了诊断消息的标准输出文本格式。其他选项说明总结在表格 1-1 在所示。

表格 1-1 在/etc/locale.conf 文件中可配置项

配置项	描述
LANG	提供系统时区的默认值
LC_COLLATE	定义该环境的排序和比较规则

LC_CTYPE	用于字符分类和字符串处理，控制所有字符的处理方式，包括字符编码，字符是单字节还是多字节，如何打印等。是最重要的一个环境变量。
LC_NUMERIC	非货币的数字显示格式
LC_TIME	时间和日期格式
LC_MESSAGES	提示信息的语言。

1.1.1.1. 显示当前配置

Localectl 命令可用于配置语言环境和键盘布局。显示当前配置，可使用如下命令：

```
# localectl status
```

1.1.1.2. 显示可用地区列表

显示可用地区列表可使用如下命令：

```
# localectl list-locales | grep en_
```

1.1.1.3. 配置地区

配置系统默认地区，需要以 root 用户身份运行：

```
# localectl set-locale LANG=zh_CN.utf8
```

用户可以配置适合的地区标示符以代替 *locale*，可通过 localectl list-locales 检索适合的地区。

1.1.2. 配置键盘布局

键盘布局配置是指文本控制台和图形用户界面的键盘布局规则。

1.1.2.1. 显示当前配置

Localectl 命令可用于配置语言环境和键盘布局。显示当前配置，可使用如下命令：

```
# localectl status
```

1.1.2.2. 显示可用键盘布局列表

显示可用地区列表可使用如下命令：

```
# localectl list-keymaps | grep cz
```

1.1.2.3. 配置键盘

配置系统默认键盘布局，需要以 root 用户身份运行：

```
# localectl set-keymap map
```

用户可以配置适合的键盘布局标示符以代替 map，可通过 localectl list-keymaps 检索适合的键盘布局。该命令还可用于配置 X11 窗口的键盘布局映射，但使用--no-convert 参数的话则不生效。同样也可用一下命令单独配置 X11 窗口的键盘布局：

```
# localectl set-x11-keymap cn
```

如果用户希望 X11 窗口和命令行终端的键盘布局不一样，可以使用如下命令：

```
# localectl --no-convert set-x11-keymap map
```

1.1.3. 其他资源

其他官方配置系统地区和键盘布局的内容可以参考安装手册。同时还可参考 1.5 获取特权章节和 3.1 章节。

1.2. 日期和时间配置

操作系统区分以下两种时区：

- 实时时间（RTC），通常作为物理时钟，它可以独立于系统当前状态计时，在主机关机情况下也可计时。
- 系统时间，是基于实时时间的由操作系统内核维护的软件时间。等系统启动内核初始化系统时间后，系统时间就独立于实时时间自行计时。

系统时间通常还保持一套世界统一时间（UTC），用于转换系统的不同时区，本地时间就是用户所在时区的真实时间。

操作系统提供了三种命令行时间管理工具，timedatectl、date 和 hwclock。以下将分别介绍各个工具的使用。

1.2.1. Timedatectl 工具使用说明

1.2.1.1. 显示当前日期和时间

命令 timedatectl 可以显示当前系统时间和机器的物理时间及其详细信息。如下示例是未启用 NTP 时钟同步的系统时间：

```
# timedatectl
```

变更 chrony 或 ntpd 服务状态不会主动通知 timedatectl 工具，如果想要更新服务的配置信息，请执行以下命令：

```
# systemctl restart system-timedated.service
```

1.2.1.2. 变更当前时间

以 root 用户运行以下命令可以修改当前时间：

```
# timedatectl set-time HH: MM: SS
```

其中 HH 代表小时，MM 代表分钟，SS 代表秒数，均需两位表示。这个命令同样可以更新系统时间和物理时间，效果类似于 date --set 和 hwclock --systohc 命令。

系统默认时间配置基于 UTC，如果想基于本地时间来配置系统时间，需要以 root 用户运行以下命令修改。

```
# timedatectl set-local-rtc boolean
```

如果基于本地时间，需要将 boolean 配置为 yes（或者 y，true，t 或者 1）。如果使用 UTC 时间，则要将 boolean 配置为 no（或者 n，false，f 或者 0）。系统默认 boolean 为 no。

1.2.1.3. 变更当前日期

以 root 用户运行以下命令可以修改当前日期：

```
# timedatectl set-time YYYY- MM- DD
```

其中 YYYY 代表年份，需 4 位数表示；MM 代表月份，需两位数表示；DD 代表日期，需两位表示。如果还需要配置时间，可以补充上时间参数，示例如下：

```
# timedatectl set-time ' 2020- 02-17 23:26:00'
```

1.2.1.4. 修改时区

执行以下命令可以显示当前时区：

```
# timedatectl list-timezones
```

以 root 用户运行以下命令可以修改当前时区：

```
# timedatectl set-timezone time_zone
```

修改时区示例如下：

```
# timedatectl list-timezones | grep Europe
```

1.2.1.5. 同步系统与远程服务器时间

以 root 用户运行以下命令可以启用/禁用时间同步服务：

```
# timedatectl set-ntp boolean
```

启用与禁用需要配置 boolean 值为 yes 或者 no。例如需要自动同步一个远程时间服务器，可以执行一下命令：

```
# timedatectl set-ntp yes
```

1.2.2. Date 工具使用说明

1.2.2.1. 显示当前日期和时间

命令 date 可以显示当前系统时间、时区、日期等信息。并可以通过参数--utc 显示当前时区时间。通过 “format”标示符来输出特定状态。常用的 format 说明如下：

表格 1-2 参数介绍

参数	描述
%H	以 HH 格式输出当前小时
%M	以 MM 格式输出当前分钟
%S	以 SS 格式输出当前秒数
%d	以 DD 格式输出当前日期
%m	以 MM 格式输出当前月份
%Y	以 YYYY 格式输出当前年份
%Z	显示时区制式，例如 C EST
%F	以 YYYY-MM-DD 格式输出当前年月日，等价于参数%Y- %m- %d
%T	以 HH:MM:SS 格式输出当前时间，等价于参数%H: %M: %S

示例如下：

```
# date
Mon Feb 17 17: 30: 24 CEST 2020
# date --utc
Mon Feb 17 15: 30: 34 UTC 2020
# date + "%Y-%m-%d%H%M"
```

```
2020- 02- 17 17: 30
```

1.2.2.2. 变更当前时间

以 root 用户运行以下命令可以修改当前时间：

```
# date --set HH: MM: SS
```

其中 HH 代表小时，MM 代表分钟，SS 代表秒数，均需两位表示。这个命令同样可以更新系统时间和物理时间，效果类似于 hwclock –systohc 命令。

系统默认时间配置基于本地时间，如果想基于 UTC 时间来配置系统时间，需要以 root 用户运行以下命令修改。

```
# date --set HH: MM: SS --utc
```

1.2.2.3. 变更当前日期

以 root 用户运行以下命令可以修改当前日期：

```
# date --set YYYY- MM- DD
```

其中 YYYY 代表年份，需 4 位数表示；MM 代表月份，需两位数表示；DD 代表日期，需两位表示。如果还需要配置时间，可以补充上时间参数，示例如下：

```
# date --set 2020-02-20 23:26 :00
```

1.2.3. hwclock 工具使用说明

1.2.3.1. 显示当前日期和时间

命令 hwclock 可以显示当前系统时间、时区、日期等信息。并可以通过参数--utc 或--localtime 显示当前 UTC 时区时间和本地时间。示例如下：

```
# hwclock
Thur 13 Feb 2020 04: 23: 46 PM CEST - 0. 329272 seconds
```

1.2.3.2. 变更当前日期和时间

以 root 用户运行以下命令可以修改当前时间：

```
#hwclock - -set - -date "dd mmm yyyy HH: MM"
```

其中 dd 代表日期 HH 代表小时，MM 代表分钟，SS 代表秒数，均需两位表示。Mmm 代表月份，以月份英文三位字母简写表示，yyyy 代表年份，以四位数字表示。这个命令通过参数--utc 或—localtime 区分配置当前 UTC 时区时间和本地时间

基于 UTC 时间来配置系统时间，需要以 root 用户运行以下命令修改，示例如下。

```
# hwclock --set --date "20 Feb 2020 21: 17" --utc
```

1.2.3.3. 同步系统与远程服务器时间

以 root 用户运行以下命令同步远程时间：

```
# hwclock - - systohc
```

1.3. 网络访问配置

1.3.1. 图形界面网络配置（ARM 和 X86）

选择【应用程序】->【系统工具】->【设置】，或者选择屏幕右上角【电源】弹出窗口选择最下面第一个【设置】按钮。进入【设置】页面，选择【网络】，进入图形化网络配置界面。

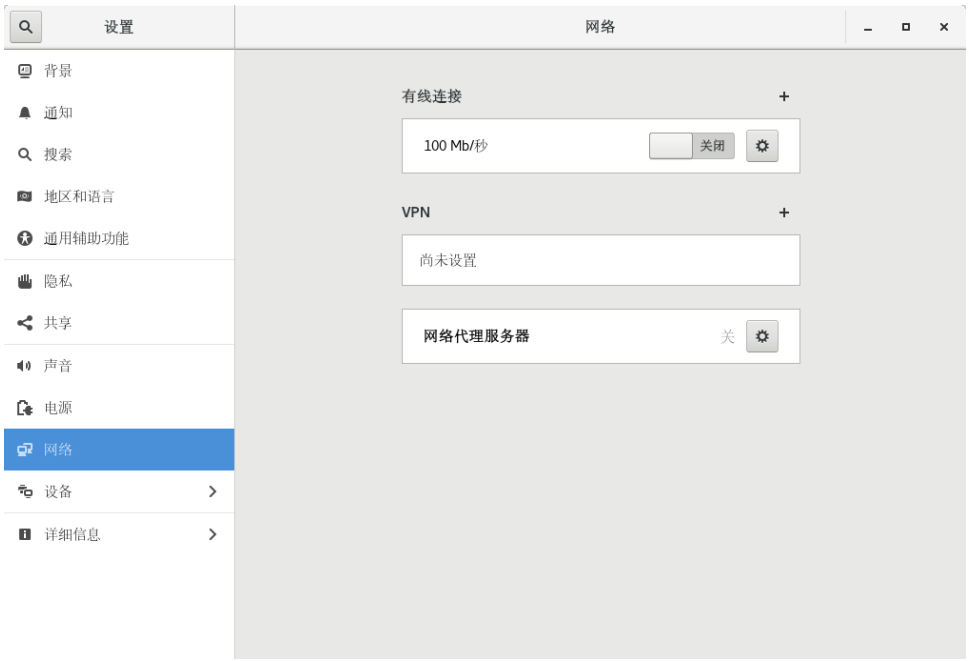


图 1-1 网络配置界面

网络配置界面包括：有线连接、VPN 和网络代理服务器。

■ **有线连接配置**

有线连接配置，可进行有线网络的打开、关闭设置，系统默认为关闭。有线网络配，具体包括“详细信息”、“身份”、“Ipv4”、“IPv6”、“安全”设置。

选择“详细信息”界面，如设置网络自动连接，请选择该项；默认选择“对其他用户可用”；默认不选择“限制后台数据使用”。未配置网路前，还会显示硬件地址；配置网络后会显示：链路速度、IPv4 地址、IPv6 地址、硬件地址、默认路由、DNS 等信息。

图 1-2 网络配置-有线-详细信息（配置网络前）

图 1-3 网络配置-有线-详细信息（配置网络后）

选择“身份”界面，可显示有线连接名称、MAC 地址和克隆的地址，可进

行 MTU 设置。

图 1-4 网络配置-有线-身份

配置 IPv4、IPv6 地址可选择“IPv4”和“IPv6 选项”。“安全”主要配置 802.1X 安全性，包括打开/关闭、认证方式、用户名、密码等。

图 1-5 网络配置-有线-IPv6

图 1-6 网络配置-有线-安全

■ **VPN**

配置本机的 VPN，如下图。

图 1-7 网络配置-VPN

■ **网络代理服务器**

配置本机的网络代理服务器，默认禁用。可配置为自动、手动方式。

图 1-8 网络配置-网络代理服务器

图 1-9 网络配置-网络代理服务器（自动和手动）

下面重点介绍 IPv4 网络图新化配置。

1.3.1.1. 动态配置 IPv4 网络

选择【应用程序】->【系统工具】->【设置】，或者选择屏幕右上角【电源】弹出窗口选择最下面第一个【设置】按钮。进入【设置】页面，选择【网络】，进入图形化网络配置界面。点击【齿轮按钮】->【IPv4】，选择【自动(DHCP)】，点击【应用】，如下界面：

图 1-10 动态网络配置

1.3.1.2. 静态网络配置

选择【应用程序】->【系统工具】->【设置】，或者选择屏幕右上角【电源】弹出窗口选择最下面第一个【设置】按钮。进入【设置】页面，选择【网络】，进入图形化网络配置界面。点击【齿轮按钮】->【IPv4】，选择【手动】，在 address 栏填写正确的 IP 地址、子网掩码、网关以及 DNS，点击【应用】。

图 1-11 静态网络配置

1.3.2. 图形界面网络配置（MIPS）

选择【应用程序】->【系统工具】->【设置】，或者选择屏幕右上角【电源】弹出窗口选择最下面第一个【设置】按钮。进入【设置】页面，选择【网络】，进入图形化网络配置界面。

图 1-12 网络配置界面

网络配置界面包括：有线连接、VPN 和网络代理服务器。

■ **有线连接配置**

有线连接配置，可进行有线网络的打开、关闭设置，系统默认为关闭。有线网络配，具体包括“详细信息”、“安全”、“身份”、“Ipv4”、“IPv6”、设置。

选择“详细信息”界面，未配置网路前，会显示硬件地址；配置网络后会显示：链路速度、IPv4 地址、IPv6 地址、硬件地址、默认路由、DNS 等信息。

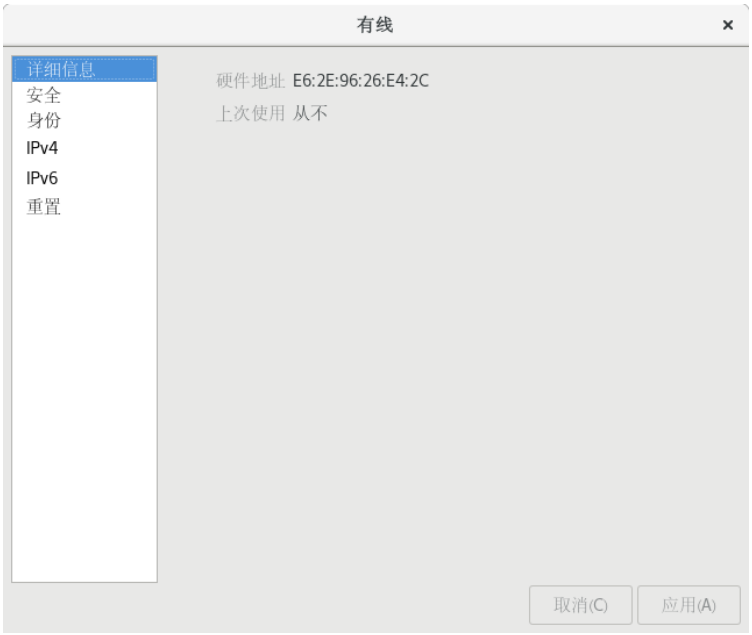


图 1-13 网络配置-有线-详细信息（配置网络前）

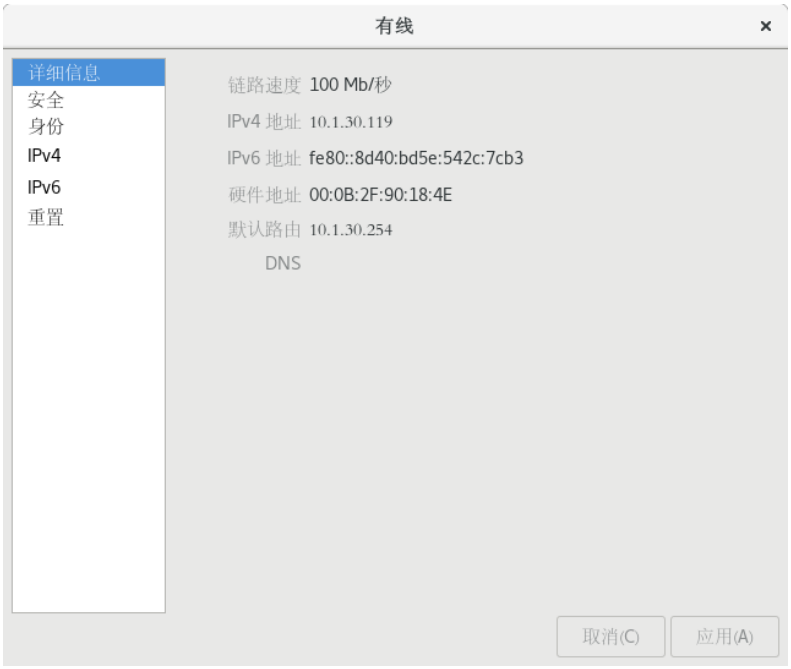


图 1-14 网络配置-有线-详细信息（配置网络后）

选择“身份”界面，可显示有线连接名称、MAC 地址和克隆的地址，可进行 MTU 设置，默认选择自动连接并对其它用户可用。

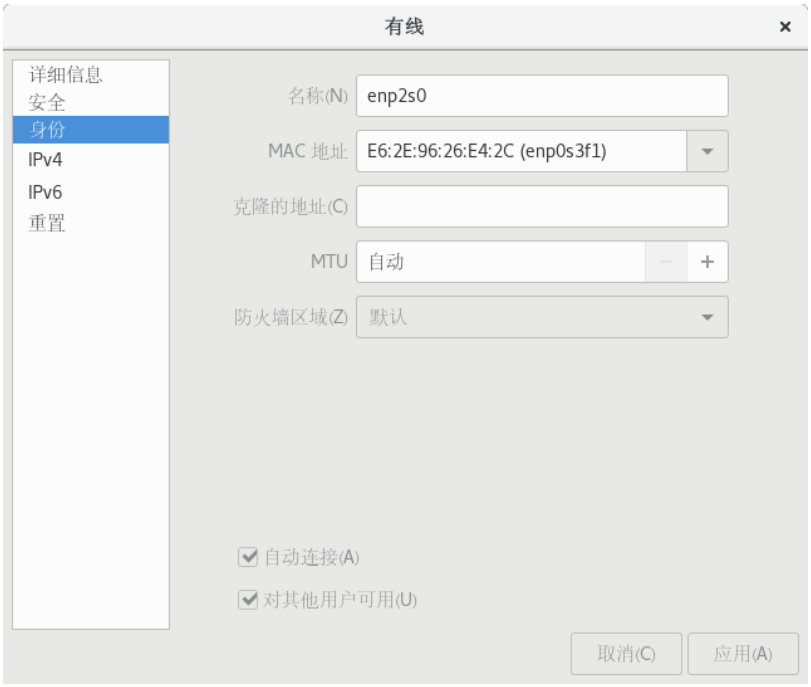


图 1-15 网络配置-有线-身份

配置 IPv4、IPv6 地址可选择“IPv4”和“IPv6 选项”。“安全”主要配置 802.1X 安全性，包括打开/关闭、认证方式、用户名、密码等。

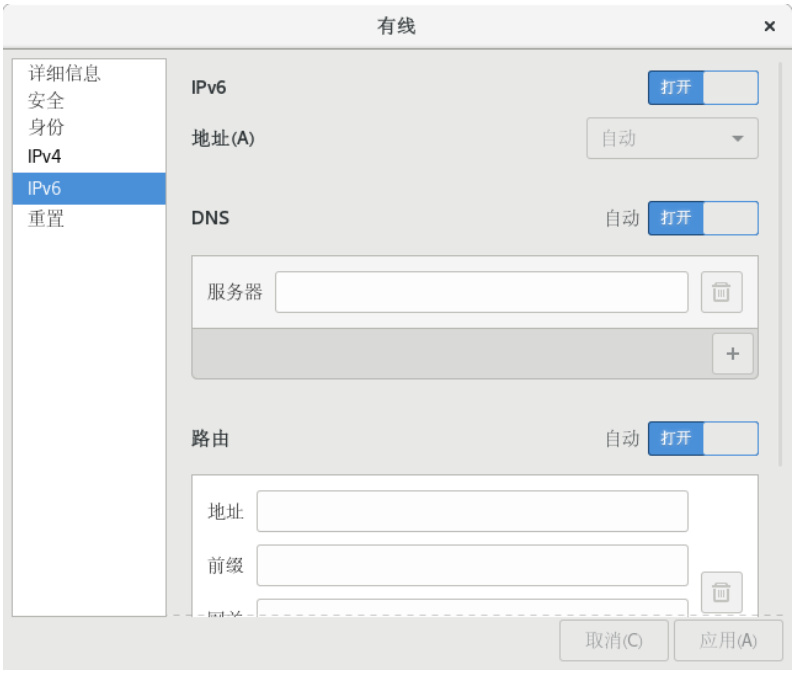


图 1-16 网络配置-有线-IPv6

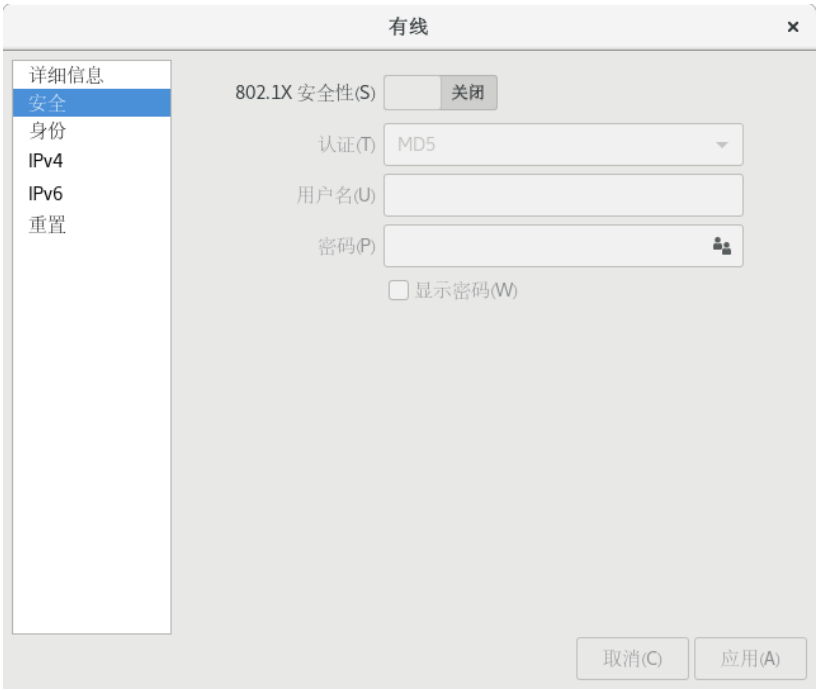


图 1-17 网络配置-有线-安全

■ **VPN**

配置本机的 VPN，如下图。

图 1-18 网络配置-VPN

■ **网络代理服务器**

配置本机的网络代理服务器，默认禁用。可配置为自动、手动方式。

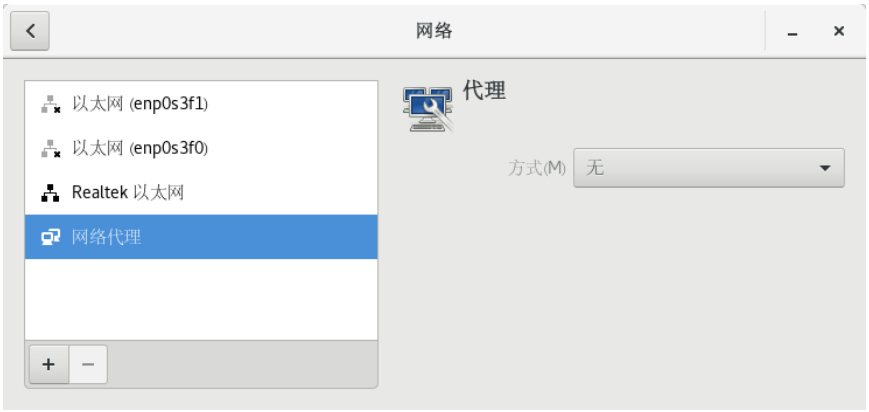
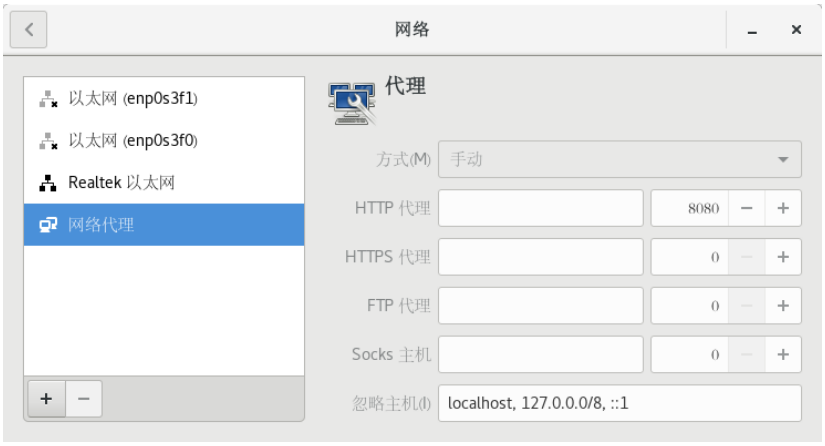


图 1-19 网络配置-网络代理服务器

图 1-20 网络配置-网络代理服务器自动



1-21 网络配置-网络代理服务器手动

下面重点介绍 IPv4 网络图新化配置。

1.3.2.1. 动态配置 IPv4 网络

选择【应用程序】->【系统工具】->【设置】，或者选择屏幕右上角【电源】弹出窗口选择最下面第一个【设置】按钮。进入【设置】页面，选择【网络】，进入图形化网络配置界面。选择要配置的网络，点击【齿轮按钮】->【IPv4】，选择【自动(DHCP)】，点击【应用】，如下界面：

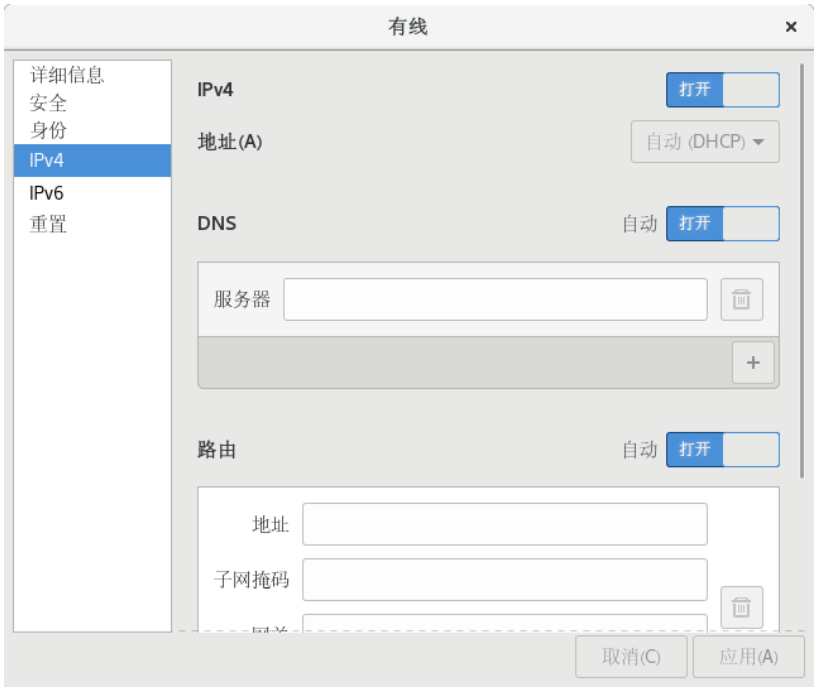


图 1-22 动态网络配置

1.3.2.2. 静态网络配置

选择【应用程序】->【系统工具】->【设置】，或者选择屏幕右上角【电源】

弹出窗口选择最下面第一个【设置】按钮。进入【设置】页面，选择【网络】，进入图形化网络配置界面。点击【齿轮按钮】->【IPv4】，选择【手动】，在 address 栏填写正确的 IP 地址、子网掩码、网关以及 DNS，点击【应用】。

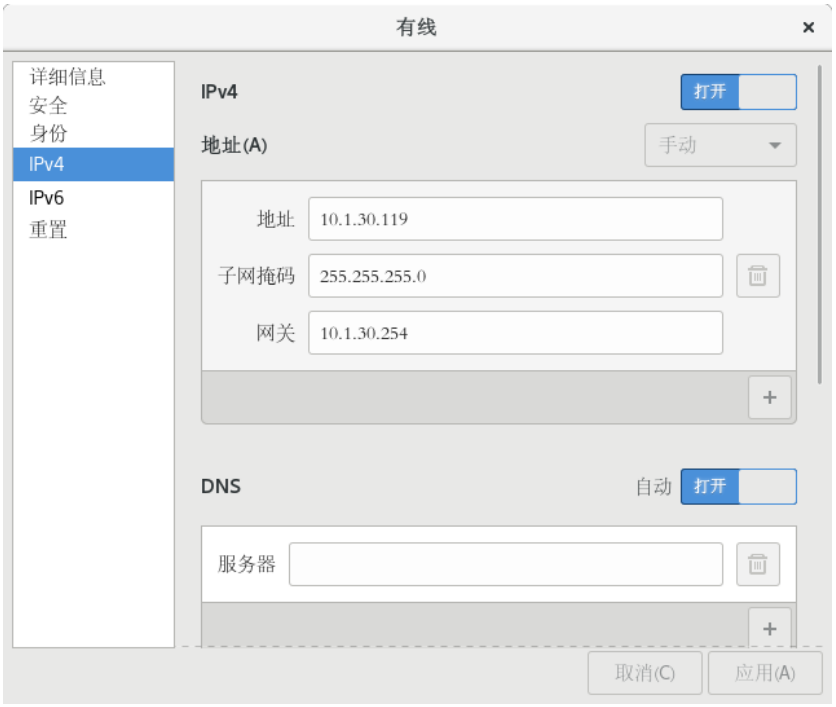


图 1-23 静态网络配置

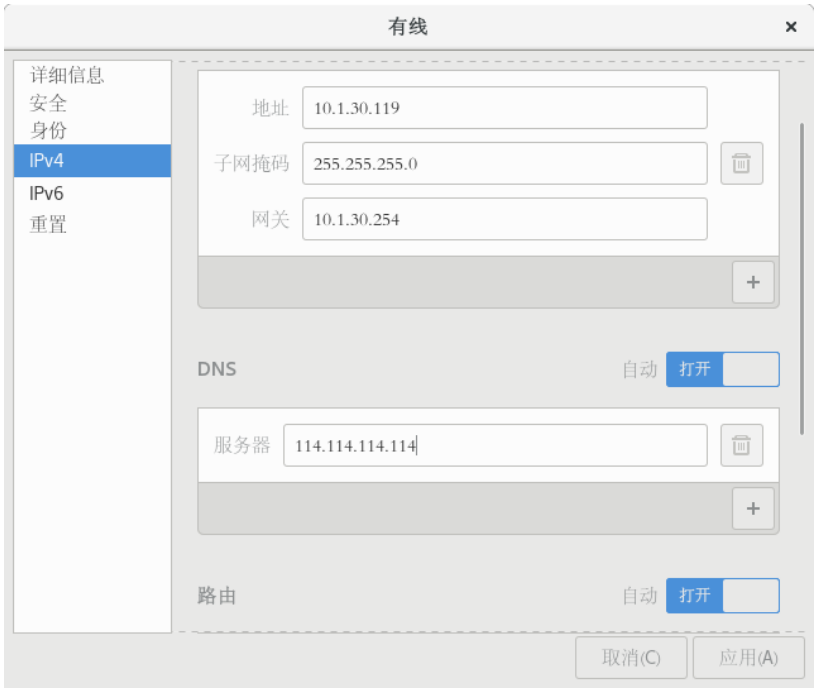


图 1-24 配置 DNS

1.3.3. 字符终端网络配置

1.3.3.1. 动态网络配置

打开终端，以网口 eth0 为例，配置文件为 ifcfg-enp1s0：

编辑文件/etc/sysconfig/network-scripts/ ifcfg-enp1s0，设置 BOOTPROTO 为 dhcp：

```
[root@localhost network-scripts]# cat ifcfg-enp1s0
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp1s0
UUID=1b862c69-046b-47c5-9550-8f41ef751237
DEVICE=enp1s0
ONBOOT=yes
HWADDR=48:5B:39:61:81:0D
DNS1=10.1.10.1
```

1.3.3.2. 静态网络配置：

打开终端，以网口 eth0 为例，配置文件为 ifcfg-enp1s0：

编辑文件/etc/sysconfig/network-scripts/ ifcfg-enp1s0，设置 BOOTPROTO 为 none，填写 IP、子网掩码、网关：

```
[root@localhost network-scripts]# cat ifcfg-enp1s0
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp1s0
UUID=1b862c69-046b-47c5-9550-8f41ef751237
DEVICE=enp1s0
ONBOOT=yes
HWADDR=48:5B:39:61:81:0D
IPADDR=10.1.30.157
PREFIX=24
GATEWAY=10.1.30.254
```

1.3.3.3. 配置 DNS：

打开终端，编辑/etc/resolv.conf，设置 nameserver：

```
# Generated by NetworkManager
nameserver 10.1.10.1
```

1.4. 用户和组群配置

用户管理者允许您查看、修改、添加和删除本地用户和组群。

1.4.1. 添加新用户（ARM 和 X86）

以下用户和组群配置内容以 ARM 和 X86 架构为例。

要使用用户管理者，您必须具备 root 特权。要从桌面启动用户管理器，点击面板上的**【应用程序】→【杂项】→【用户和组群】**，弹出用户配置对话框。

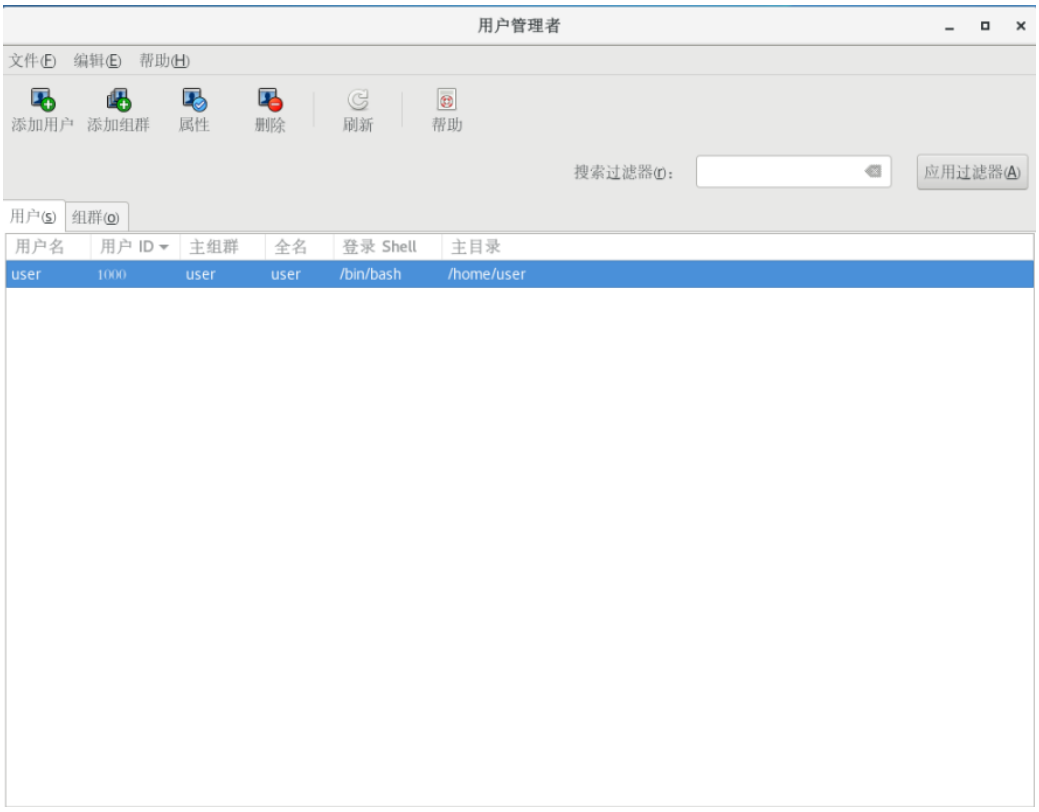


图 1-25 用户管理（一）

双击用户列表中用户信息，弹出【用户属性】对话框，可编辑用户数据、帐号信息、密码信息和组群各项参数。

- 用户数据：显示所添加用户的基本配置信息，可进行修改用户的全名，密码，主目录及登录 shell。
- 帐号信息：选择启用账户过期能够将用户帐户设置为某个时间过期，在响应的地方输入时间即可。选择锁定本地密码可以锁定用户帐户，阻止该用户登入系统。
- 密码信息：显示用户上次更改密码的时间。要过一段时间后强制用户更改密码，选择启用密码过期，然后在密码有效时间中输入对应值即可。用户密码过期的天数，提示用户修改密码前所过的天数，以及帐户在停用多少天内认可修改。
- 组群：允许查看配置用户主要所在组，以及设置其他想让该用户加入的组。

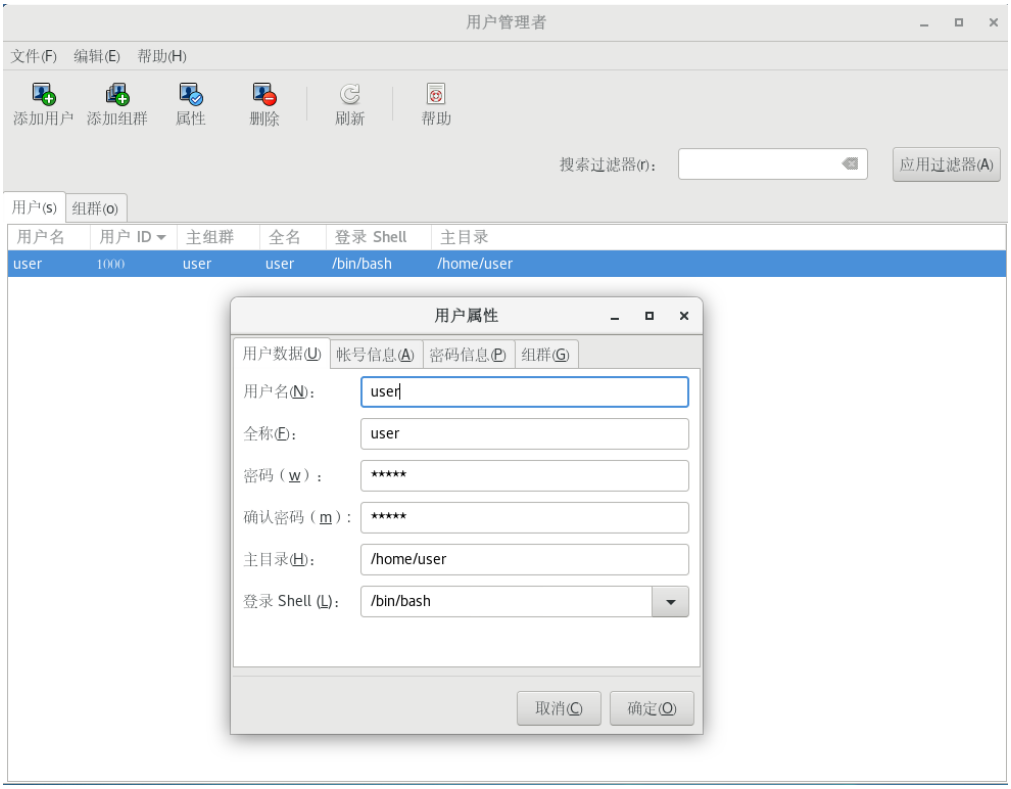


图 1-26 用户管理（二）

要添加新用户，点击**【添加用户】**按钮。如图所示的窗口就会出现。在适当的字段内键入新用户的用户名和全名。密码提示至少包含八个字符。

【用户名】会配置用户主目录，默认的主目录是 home/用户名/。您可以改变为用户创建的主目录。点击**【添加】**来创建该用户。

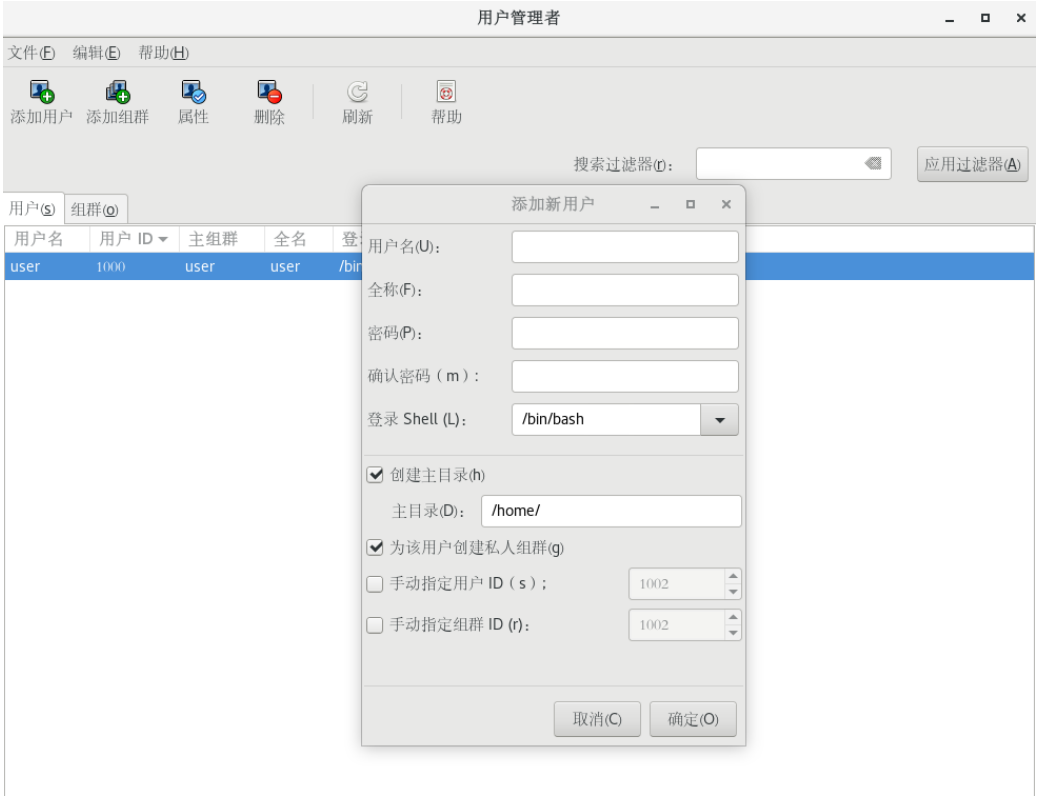


图 1-27 创建新用户

1.4.2. 添加新用户（MIPS）

以下用户和组群配置内容以龙芯 MIPS 架构为例。

要使用用户管理者，您必须具备 root 特权。要从桌面启动用户管理器，点击面板上的**【应用程序】→【系统工具】→【设置】→【用户】**，弹出用户配置对话框。

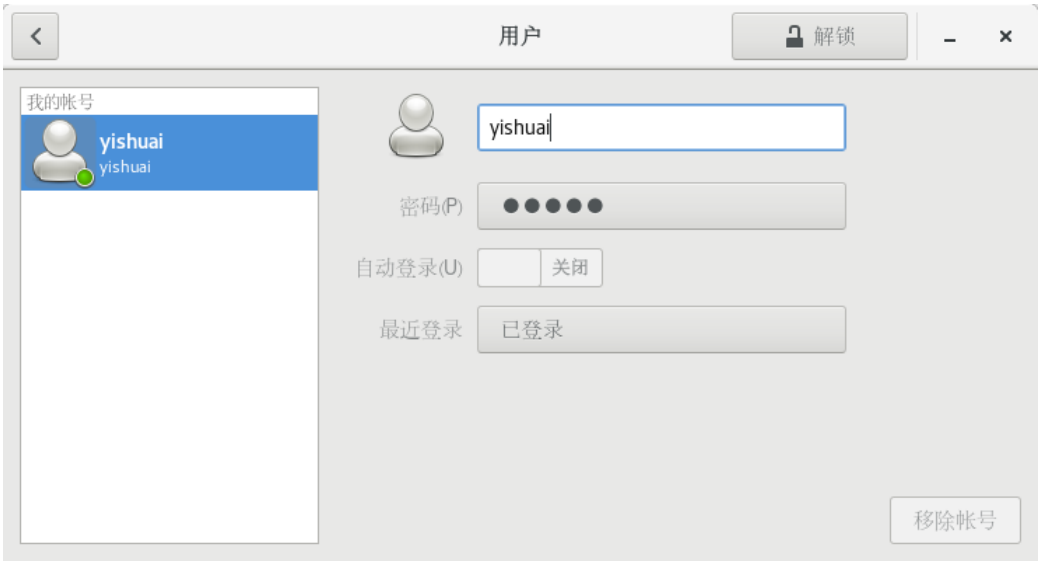


图 1-28 用户管理（一）

点击【**解锁**】，弹出验证对话框输入账号正确密码后，可编辑各项参数及添加用户。

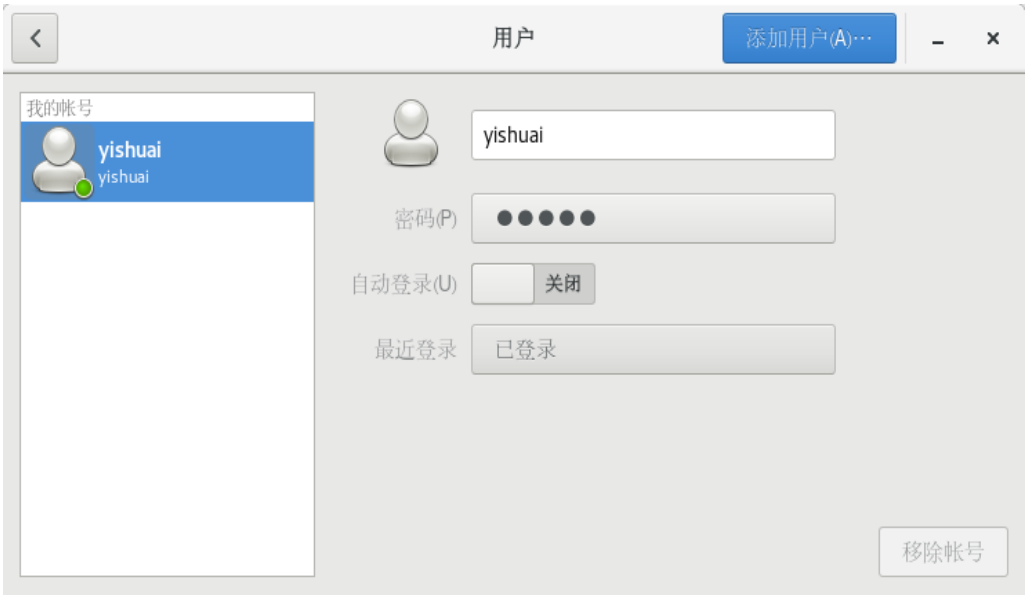


图 1-29 用户管理（二）

要添加新用户，解锁后点击【**添加用户（A）...**】按钮。如图所示的窗口就会出现。在适当的字段内键入新用户的用户名和全名。【**账户类型**】可以配置管理员权限和标准用户权限。在【**密码**】字段可选择“用户下次登录修改密码”或“现在设置密码”。密码必须至少包含八个字符。

【用户名】会配置用户主目录，默认的主目录是 home/用户名/。您可以改变为用户创建的主目录。点击【添加】来创建该用户。

图 1-30 创建新用户

修改用户属性

要查看某个现存用户的属性，点击【用户】标签，从用户列表中选择该用户，右侧可现实用户的详细信息，并可在右侧编辑这些属性。

【账户类型】区分标准用户还是管理员用户。

【语言】用户登录默认语言选项。

【密码】用户登录密码，单击即可进行修改配置。

【最近登录】选显示该用户最近一次登录系统的时间。

1.4.3. 命令行配置

如果您更喜欢使用命令行工具，请参考本节来配置用户和组群。

表格 1-3 常用的命令行

选项	描述
id	显示用户和组群 id
useradd,usermod,userdel	添加、修改、删除用户
Groupadd.groupmod,	添加、修改、删除组群

选项	描述
groupdel	
gpasswd	修改/etc/group 配置文件
pwck, grpck	检查用户和组群密码文件完整性。
pwconv, pwunconv	开/关用户影子密码
grpconv, grpunconv	开/关组群影子密码

1.4.3.1. 添加用户

要在系统上添加用户：

使用 useradd 命令来创建一个锁定的用户账号：

```
useradd <username>
```

使用 passwd 命令，通过指派密码和密码过期规则来给某账号解锁：

```
passwd <username>
```

useradd 的命令行选项在表格 1-4 中被列出。

表格 1-4 useradd 命令行选项

选项	描述
-c comment	用户的注释
-d home-dir	用来取代默认的 /home/username/ 主目录
-e date	禁用账号的日期，格式为：YYYY-MM-DD
-f days	密码过期后，账号被禁用前要经过的天数（若指定了 0，账号在密码过期后会被立刻禁用。若指定了-1，密码过期后，账号将不会被禁用）
-g group-name	用户默认组群的组群名或组群号码（该组群在指定前必须存在）
-G group-list	用户是其中成员的额外组群名或组群号码（默认以外的）的列表，用逗号分隔（组群在指定前必须存在）
-m	若主目录不存在则创建它
-M	不要创建主目录
-n	不要为用户创建用户私人组群
-r	创建一个 UID 小于 500 的不带主目录的系统账号
-p password	使用 crypt 加密的密码
-s	用户的登录 shell，默认为 /bin/bash
-u uid	用户的 UID，它必须是独特的，且大于 499

1.4.3.2. 添加组群

要给系统添加组群，使用 groupadd 命令：

```
groupadd <group-name>
```

groupadd 的命令行选项在表格 1-5 中被列出。

表格 1-5 groupadd 命令行选项

选项	描述
-g gid	组群的 GID，它必须是独特的，且大于 499
-r	创建小于 500 的系统组群
-f	若组群已存在，退出并显示错误（组群信息不会被改变）。若指定了 -g 和 -f 选项，但是组群已存在，-g 选项就会被忽略

1.4.4. 添加用户的详细过程

下列步骤演示了在启用屏蔽密码的系统上使用 useradd juan 命令后的情形：

1) 在/etc/passwd 文件中新添了有关 juan 的一行。这一行的特点如下：

a） 它以用户名 juan 开头。

b） 密码字段有一个“x”，表示系统使用屏蔽密码。

c） 500 或 500 以上的 UID 被创建。（在中标麒麟服务器操作系统中，500 以下的 UID 和 GID 被保留给系统使用。）

d） 500 或 500 以上的 GID 被创建。

e） 可选的 GECOS 信息被留为空白。

f） juan 的主目录被设为 /home/juan/。

g） 默认的 shell 被设为 /bin/bash。

2) 在 /etc/shadow 文件中新添了有关 juan 的一行。这一行的特点如下：

a） 它以用户名 juan 开头。

b） 出现在 /etc/shadow 文件中密码字段内的两个叹号（!!）会锁住账号。

c） 如果某个加密的密码使用了 -p 选项被传递，这个密码会被放置在/etc/shadow 文件中用于该用户的那一行中，密码被设置为永不过期。

3) 在 /etc/group 文件中新添了一行有关 juan 组群的信息。和用户名相同的组群叫做用户私人组群（user private group）。在 /etc/group 文件中

新添的这一行具有如下特点：

a） 它以组群名 juan 开头。

b） 密码字段有一个“x”，表示系统使用屏蔽密码。

c） GID 与列举 /etc/passwd 文件中用户 juan 行中的相同。

4) 在 /etc/gshadow 文件中新添了有关 juan 组群的一行。这一行的特点如下：

a） 它以组群名 juan 开头。

b） 出现在 /etc/gshadow 文件中密码字段内的一个叹号（!）会锁住该组群。

c） 所有其它字段均为空白。

5) 用于用户 juan 的目录被创建在 /home/ 目录之下。该目录为用户 juan 和组群 juan 所有。它的读写和执行权限仅为用户 juan 所有。所有其它权限都被拒绝。

6) /etc/skel/ 目录（包含默认用户设置）内的文件被复制到新建的 /home/juan/ 目录中。

这时候，系统上就存在了一个叫做 juan 的被锁的账号。要激活它，管理员必须使用 passwd 命令给账号设置一个密码，它还可以设置密码过期规则。

1.5. 获取特权

系统普通用户的权限有不同的限制，某些情况下普通需用需要执行管理员用户权限才能执行的命令，此时可以通过 su 或者 sudo 命令获得管理员权限特权。

1.5.1. Su 命令工具

用户使用 su 命令时，需要输入 root 用户密码，验证通过后可以获取 root 的脚本环境。一旦通过 su 命令登入，这个用户的所有操作均视为 root 用户操作。由于 su 可以获取 root 全部权限，并因此获取其他用户的权限，可能存在一定安全问题。因此可以通过管理员组群 wheel 来进行限制。以 root 用户执行以下命令：

```
# usermod -G wheel username
```

将用户加入 wheel 组群后，可以限制只有这个组群的用户可以使用 su 命令访问。配置 su 的 PAM 可以编辑/etc/pam. d /su 文件，通过添加删除#字符来确认添加或删除相应内容。

```
#auth required pam_wheel. so use_uid
```

上述内容表示管理员组群 wheel 内的用户可以通过 su 访问其他用户。

1.5.2. Sudo 命令工具

Sudo 命令允许系统管理员让普通用户执行一些或者全部的 root 命令。当可信用户执行 sudo 命令时，需要提供他们自己的用户密码，然后以 root 权限执行命令。

基本的 sudo 命令如下：

```
#sudo  command
```

Sudo 命令有很大的弹性，只有在/etc/sudoers 文件中被允许的用户可以执行在他们自己的 shell 环境中执行 sudo 命令，而不是 root 的 shell 环境。这意味着在 7 系列中 root 的 shell 环境是被禁止访问的。

配置 sudo 必须通过编辑/etc/sudoers 文件，而且只有管理员用户才可以修改它，必须使用 visudo 编辑。之所以使用 visudo 有两个原因，一是它能够防止两个用户同时修改它；二是它也能进行一些的语法检查。以 root 身份用 visudo 打开配置文件，输入以下内容：

```
#juan    ALL=(ALL)       ALL
```

这条信息意思是 juan 用户可以以任何主机连接并通过 sudo 执行任何命令。

下面这条信息说明 users 用户可以本地主机可以执行/sbin/shutdown - h now 命令：

```
%users  localhost=/sbin/shutdown  -  h  now
```

1.6. GB18030 支持说明

系统默认安装情况下，GB18030 的用户自定义区中部分码位被以下字体占用：cjkuni-uming、jomolhari、paktype-naskh-basic、sil-abyssinica、ucs-miscfixed、overpass、stix、vlgothic、wqy-zenhei、fangzheng-song/FZFSJW.TTF。

如需要使用用户自定义区码位，请将以上字体酌情删除，如有疑问，直接联系中标软件有限公司技术支持。

第二章 YUM 安装和管理软件

Yum 是基于 RPM 包的前端软件包管理器，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软件包，无须繁琐地一次次下载、安装。

用户可以创建、添加和删除软件包安装源，yum 可以覆盖 rpm 工具的全部功能，并可通过简洁命令优化功能。Yum 还可以简便的管理在本机上安装和下载的

rpm 包，也可利用自己的特性下载软件包。

2.1. 检查和升级软件包

2.1.1. 软件包升级检查

查看系统里已经安装的软件包有哪些可以升级可以执行以下命令，以 X86 平台示例如下：

```
# yum check-update
```

示例说明：

- PackageKi t——软件包名称；
- x86_64——该软件包支持的 CPU 架构；
- 0.5.8——可升级的软件包版本
- neokyin——可升级的软件包所存储仓库

上述输出可以看出还可升级内核 kernel，yum 和 rpm 包本身以及它们的依赖关系。

2.1.2. 升级软件包

Yum 支持一次升级单个/批量软件包，并同时安装/更新相应的依赖包。

1. 升级单一软件包命令：

```
# yum update package_name
```

升级 rpm 软件包命令：

```
#yum update rpm
```

上述输出的说明如下：

a) Loaded plugins: langpacks, product- id——显示已安装和可用的 yum p lug-ins 信息。

b) rpm. x86 _6 4：用户需要下载升级的软件包和依赖软件包。

c) Yum 默认会显示升级软件包的基本信息，并提示是否确认安装，用户可以在使用 yum 命令是添加参数 –y，效果等同于出现 Is this ok [y/d/N]: 时输入 yes。输入 d 则进行软件包下载。

d) 安装过程中如果出现错误导致安装过程终止，可以使用 yum history 命令查看详细描述。

如果需要安装一组软件包，可以以 root 用户执行命令：

```
# yum group update group_name
```

2. 批量升级软件包及其依赖

如果需要升级系统所有软件包，可以使用以下命令：

```
# yum update
```

2.1.3. 利用系统光盘与 yum 离线升级系统

当系统处于离线状态或者无法访问官方更新源时，可以利用更新的系统光盘创建本地 yum 源并进行升级。步骤如下：

1. 创建系统光盘挂载目录，以 root 用户执行：

```
# mkdir mount_dir
```

2. 将系统安装光盘挂载至该目录，以 root 用户执行

```
# mount -o loop iso_name mount_dir
```

3. 将系统光盘中的 media.repo 文件从挂载目录拷贝至/etc/yum.repo s.d/目录下：

```
# cp mount_dir/med ia. repo /etc/yum. repos.d/new. repo
```

4. 编辑/etc/yum. repos.d/new. repo 配置文件以添加光盘路径：

```
# baseurl=file://mount_dir
```

5. 更新 yum 源并进行升级，以 root 用户执行：

```
# yum update
```

6. 升级成功后，卸载系统光盘挂载目录：

```
# umount mount_dir 或者 rmdir mount_dir
```

如果不再使用这个 yum 源进行安装和升级，可以以 root 用户删除文件：

```
# rm /etc/yum. repos. d /new. repo
```

2.2. 管理软件包

Yum 提供了完整操作系统软件包管理功能，包括检索、查看信息、安装和删除。

2.2.1. 检索软件包

执行 yum search 命令可以检索软件包，例如检索包含“meld”和“kompare”字段的软件包，以 X86 平台示例如下：

```
# yum search meld kompare
```

如果 yum 检测的结果繁多，可以通过 shell 本身的 grep 或者正则表达式进行过滤。

2.2.2. 安装包列表

显示已安装和可安装的软件包列表可以执行以下命令：

```
# yum list all
```

显示包括某些字符的已安装和可安装软件包列表可以执行以下命令：

```
# yum list glob_expression…
```

显示 abrt 相关软件包列表的命令如下：

```
# yum list abrt- addon\* abrt- plugin\*
```

显示包括某些字符的已安装软件包列表可以执行以下命令：

```
# yum list installed glob_expression…
```

显示包括 krb 的所有已安装软件包示例如下：

```
# yum list installed "krb?- *"
```

显示包括某些字符的可安装软件包列表可以执行以下命令：

```
# yum list available glob_expression…
```

显示所有可用的 gstreamer plug-ins 软件包列表：

```
# yum list available gstreamer\*plugin\*
```

查看软件仓库

成功注册后，可使用 yum 来管理软件包。

查看可用的软件仓库可以使用以下命令：

```
# yum repolist
```

如果想显示更多信息可以加上-v 选项，或者用 yum repoinfo 命令输出信息。

```
# yum repolist -v
```

```
# yum repoinfo
```

如果需要显示所有可用和不可用的软件仓库，可以使用以下命令：

```
# yum repolist all
```

2.2.3. 显示软件包信息

显示一个或多个软件包可以使用以下命令：

```
# yum info package_name…
```

显示软件包 abrt 详细信息的命令：

```
# yum info abrt
```

用户还可以通过查询 yum 数据库查询软件包相关信息：

```
# yumdb info package_name
```

显示软件包 yum 详细信息的命令：

```
# yum info yum
```

2.2.4. 安装软件包

用户可以以 root 用户使用以下命令安装软件包

```
# yum install package_name
```

安装 sqlite 的 i686 架构的软件包示例：

```
# yum install sqlite.i686
```

除了安装软件包，还可以安装具体的二进制文件，您可以输入文件地址，以 root 用户执行安装：

```
# yum install /usr/sbin/named
```

安装命令如下：

```
# yum install httpd
```

如果要安装本地软件包，可以执行：

```
# yum localinstall path
```

2.2.5. 下载软件包

在执行安装流程中，显示以下选项是：

```
. . .
Total size: 1. 2 M
Is this ok [y/d/N]:
. . .
```

输入 d，可以执行软件包下载。软件包默认下载路径为/var/cache/yum/$base arch/$releasever/packages/。

2.2.6. 删除软件包

删除软件包可以执行以下命令：

```
yum remove package_name…
```

删除 totem 软件包示例：

```
yum remove totem
```

2.3. 管理软件包组

软件包组可以搜集一系列特定功能软件包，比如系统工具和视频软件包组。安装软件包组可以一起安装其依赖。

2.3.1. 软件包组列表

Summary 选项可以显示软件包组的基本信息：

```
# yum groups summary
```

以下为输出示例：

```
# yum groups summary
```

显示某个软件包组的全部信息可以用以下命令：

```
# yum groups info glob_expression…
```

以下为办公软件包组输出示例：

```
# yum group info LibreOffice
```

说明如下：

“-”软件包没有安装并且也不会作为包组内容安装。

“+”软件包没有安装但是进行软件包或包组升级时将会被安装。

“=”软件包已经安装且作为包组的一部分。

No symbol：软件包已经安装，且不属于包组。

2.3.2. 安装软件包组

每个软件包组都有自己的组 ID，要显示包组 id 可以使用以下命令：

```
# yum group list ids
```

查找 KDE 桌面软件包组列表的示例：

```
# yum group list ids kde\*
```

有些包组是作为因此软件仓库的，例如，使用因此命令选项示例：

```
# yum group list hidden ids kde\*
```

软件包组的安装可以通过软件包组名称安装，也可通过包组 id 安装。

```
# yum group install "group name"
# yum group install groupid
```

也可用通过以下两种命令安装：

```
# yum install @group
# yum install @^group
```

下面是 4 中安装 KDE 桌面软件分组的示例：

```
# yum group install "KDE Desktop"
# yum group install kde-desktop
# yum install @"KDE Desktop"
# yum install @kde-desktop
```

2.3.3. 删除软件包组

可以通过软件包组名或者软件包组 id 删除软件包。以 root 权限执行：

```
# yum group remove group_name
# yum group remove groupid
```

如果软件分组有@标签，也可用以下命令删除。以 root 身份执行：

```
# yum remove @group
# yum remove @^group
```

删除 KDE 桌面软件分组示例：

```
# yum group remove "KDE Desktop"
# yum group remove kde- desktop
# yum remove @"KDE Desktop"
# yum remove @kde-desktop
```

2.4. 软件包操作记录管理

Yum 所有的操作记录默认存在放在/var/lib/yum/history/目录，并可以使用 yum history 命令进行管理操作。

2.4.1. 查看操作

显示以往 20 条 yum 操作记录，可以使用以下命令。以 root 权限执行：

```
# yum history list
```

如果想显示所有 yum 操作记录，可以使用以下命令。以 root 权限执行：

```
# yum history list all
```

如果想显示某一部分 yum 操作记录，可以使用以下命令。以 root 权限执行：

```
# yum history list start_id. . end_id
```

显示过去 5 条 yum 信息示例如下：

```
[root@localhost ~]# yum history list 1..5
```

以上 yum history list 输出显示内容说明如下：

ID——识别特定记录的标示数；

Login user——区别 yum 命令的操作用户；

Date and time——该条记录的日期和时间；

Action(s)——操作的具体内容；

Altered——记录操作影响的条目数；

下表是 Action 的不同说明：

表格 2-1 Action 说明

Action	缩写	描述
Downgrade	D	下载更新包
Erase	E	删除软件包
Install	I	安装软件包

Obsoleting	O	软件包标注废弃
Reinstall	R	软件包重装
Update	U	升级软件包

如果想要同步 rpmdb 或者 yumdb 数据库，可以使用以下命令：

```
# yum history sync
```

如果想显示 yum 历史及数据库状态信息可以使用以下命令：

```
# yum history stats
```

输出 yum history stats 示例如下：

```
[root@localhost ~]# yum history stats
```

Yum 还支持提供过去记录的总结信息，以 root 身份执行以下信息：

```
# yum history summary
```

如果只想查看某一阶段的记录总结信息，以 root 身份执行以下信息：

```
# yum history summary start_id. . end_id
```

显示最后 5 条记录总结信息示例如下：

```
# yum history summary 1. . 5
```

Yum 记录显示还支持通过软件包进行查找，命令如下：

```
# yum history package- list glob_expression…
```

2.4.2. 审查操作

需要显示某条操作记录的具体综述信息，可以执行以下命令：

```
# yum history summary id
```

其中 id 是操作的的 id。

如果需要显示某条操作记录的详细信息，可以使用以下命令：

```
# yum history info id
```

如果需要显示某一阶段操作记录的详细信息，可以使用以下命令：

```
# yum history info start_id. . end_id
```

示例如下：

```
# yum history info 4 . . 5
```

同样的，用户还可以查询附件信息，命令如下：

```
# yum history addon-info id
```

以及显示最后的记录附加信息命令：

```
# yum history addon-info last
```

示例如下：

```
# yum history addon-info 4
```

2.4.3. 恢复与重复操作

如果想要撤销某个 yum 操作，可以以 root 权限执行一下操作：

```
# yum history undo id
```

如果需要重复某个 yum 操作，可以以 root 权限执行一下操作：

```
# yum history redo id
```

2.4.4. 启用新的操作历史

Yum 的操作记录存为一个单独的 SQLite 数据文件。如果需要启用新的操作记录，可以以 root 权限执行命令：

```
# yum history new
```

这个命令会在/var/l ib/yum/history/目录创建新的数据库文件，旧的数据库文件也会保留。

2.5. 配置 yum 和 yum 仓库

Yum 的配置文件为/etc/yum. conf。其中【main】字段是必填选项，用于配置 yum 的基本规则。它也包含放在/etc/yum. repo s.d/目录的.repo 文件作为仓库。

2.5.1. 配置【main】选项

```
# [main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
```

```
logfile=/var/log/yum. log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=3
[comments abridged]
# PUT YOUR REPOS HERE OR IN separate files named file. rep
o
# in /etc/yum. repos. d
```

各字段说明如下：

Cachedir：yum 缓存的目录，yum 在此存储下载的 rpm 包和数据库，默认是/var/cache/yum。

Debuglevel：除错级别，0-10,默认是 2。

Exactarch：有两个选项 1 和 0,代表是否只升级和当前安装软件包 cpu 体系一致的包，如果设为 1，则如您安装了一个 i386 的 rpm，则 yum 不会用 i686 的包来升级。

Gpgchkeck：有 1 和 0 两个选择，分别代表是否进行 gpg 校验，如果没有这一项，默认是检查的。

Keepcache：缓存是否保存，1 保存，0 不保存。

Logfile：yum 的日志文件，默认是/var/log/yum.log。

Obsoletes：这是一个 update 的参数，相当于 upgrade，允许更新陈旧的 RPM 包。默认为 1 启动更新。

Pkgpolicy： 包的策略。一共有两个选项，newest 和 last，这个作用是如果您设置了多个 repository，而同一软件在不同的 repository 中同时存 在，yum 应该安装哪一个，如果是 newest，则 yum 会安装最新版本。如果是 last，则 yum 会将服务器 id 以字母表排序，并选择最后的那个服务器上的软件安装。一般都是选 newest。

distroverpkg：指定一个软件包，yum 会根据这个包判断您的发行版本，默认是 kylin-release，也可以是安装的任何针对自己发行版的 rpm 包。

Retries：网络连接发生错误后的重试次数，如果设为 0，则会无限重试。

2.5.2. 配置【repository】选项

```
[repository]
name=repository_name
baseurl=repository_url
```

每一个仓库选项【repository】都必须包括以下内容：

name=repository_name：软件源的名称，将被 yum 获取并识别。

baseurl = repository_url：是服务器设置中最重要的部分，只有设置正确，才能从上面获取软件。它的格式是：

baseurl=url://server1/path/to/repository/

url://server2/path/to/repository/

url://server3/path/to/repository/

其中 url 支持的协议有 http:// ftp:// file://三种。baseurl 后可以跟多个 url，您可以自己改为速度比较快的镜像站，但 baseurl 只能有一个，也就是说不能像如下格式：

baseurl=url://server1/path/to/repository/

baseurl=url://server2/path/to/repository/

baseurl=url://server3/path/to/repository/

其中 url 指向的目录必须是这个 repository header 目录的上一级，它也支持 $releasever $basearch 这样的变量。

enabled= value：这个选项表示这个 repo 中定义的源是否启用的，0 为禁用，1 为启用。

gpgcheck= value：这个选项表示这个 repo 中下载的 rpm 将进行 gpg 的校验，已确定 rpm 包的来源是有效和安全的。1 为启用，0 为禁用。

以下为一个 repo 文件示例：

```
[ns10-adv-x64-os]
name=KYLIN Linux Advanced Server V7.0 - Os
baseurl=http://IP-address /NS/V7.0Update2/os/adv/$basearch/
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin-release
enabled=1
```

```
[ns10-adv-x64-updates]
name=KYLIN Linux Advanced Server V7.0 - Updates
baseurl=http://IP-address /NS/V7.0Update2/os/updates/adv/$basearch/
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin-release
enabled=0

[ns10-adv-x64-addons]
name=KYLIN Linux Advanced Server V7.0 - Addons
baseurl=http:// IP-address /NS/V7.0Update2/addons/adv/$basearch/
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin-release
```

2.5.3. 使用 Yum 变量

配置 yum 文件也可使用变量参数，说明如下：

$releasever：发行版的版本，从[main]部分的 distroverpkg 获取，如果没有，则根据 neokylin-release 包进行判断。

$arch：cpu 体系结构，如 i586，i686，x86_64 等

$basearch，cpu 的基本体系组，如 i686 和 athlon 同属 i386，alpha 和 alphaev6 同属 alpha。

2.5.4. 查看当前配置

查看 yum 当前所有配置，可以执行以下命令：

```
yum-config-manager
```

如果只想查看某个或某些字段，可以执行以下命令：

```
yum-config-manager section…
```

如果想查看匹配某些配置的字段，可以执行以下命令：

```
yum-config-manager glob_expression…
```

以下是显示【main】字段配置信息的示例：

```
#yum-confi g - manager main \*
```

```
Loaded plugins: langpacks, product- id
========================= main====================
[main]
alwaysprompt = True
assumeyes = False
bandwith = 0
bugtracker_url = http://bugzilla.kylinos.cn/bugzilla
cache = 0
[output truncated]
```

2.5.5. 添加、启用和禁用 yum 软件仓库

Yum 支持用户添加软件仓库，添加软件仓库的命令如下，以 root 身份执行：

```
yum-config-manager --add-repo repository_url
```

其中 repository_url 是.repo 文件的链接。

下边是添加 http://www.example.com/example.repo 软件仓库的示例：

```
#yum-config-manager --add-repo http://www. example.com/example.r
epo
Loaded plugins: langpacks, product- id
adding repo from: http: //www. example. com/example. repo
grabbing file http: //www. example. com/example. repo to
/etc/yum. repos. d/example. repo
example. repo | 413 B
00: 00
repo saved to /etc/yum. repos. d/example. repo
```

启用 yum 软件仓库命令如下，以 root 权限执行：

```
# yum-config-manager --enable repository…
```

如果只想启用某一个字段，可以以 root 权限执行：

```
# yum-config-manager --enable glob_expression…
```

例如，启用在/etc/yum.conf 中配置的【example】【example-debuginfo】和【e

xamplesource】字段。

```
# yum-config-manager--enable example\*
Loaded plugins: langpacks, product- id
============ repo: example====================
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = http: //www. example. com/repo/7Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/example
[output truncated]
```

如果想启用/etc/yum.repos.d /目录下的所有软件仓库和/etc/yum.conf 文件中的仓库，示例如下：

```
# yum-config-manager --enable \*
Loaded plugins: langpacks, product- id
====================== repo: example================
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = http: //www. example. com/repo/7Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/example
[output truncated]
```

禁用 yum 软件仓库

禁用软件仓库命令如下，以 root 权限执行：

```
# yum-config-manager --disable repository…
```

如果只想启用某一个字段，可以以 root 权限执行：

```
# yum-config-manager --enable glob_expression…
```

如果想禁用/etc/yum.repos.d /目录下的所有软件仓库和/etc/yum.conf 文件中的仓库，示例如下：

```
# yum-config-manager --disable \*
Loaded plugins: langpacks, product- id
================ repo: example======================
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = http: //www. example. com/repo/7Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/example
[output truncated]
```

2.5.6. 创建 yum 软件仓库

创建软件仓库步骤如下：

1. 安装 createrepo 软件包：

```
# yum install createrepo
```

将想要创建仓库的所有软件包复制到一个目录，例如/mnt/local_repo/。

2. 创建软件仓库

```
# createrepo --database /mnt/local _repo
```

2.6. Yum 插件

查看 yum 插件命令如下：

```
# yum info yum
Loaded plugins: langpacks, product- id
[output truncated]
```

2.6.1. 启用、配置和禁用 yum 插件

启用 yum 插件需要修改/etc/yum.conf 文件的【main】字段，确保 plugins 的值为一：

```
# plugins=1
```

如果禁用则赋值为 0：

```
# plugins=0
```

每个已安装插件都有自己的配置文件目录/etc/yum/pluginconf.d/，以下是 aliases 插件的配置文件 aliases.conf 示例：

```
[main]
enabled=1
```

如果想使用一条命令禁用一个或多个插件，可以通过参数- -d isableplugin=plugin_name 实现。例如禁用 aliases 插件如下：

```
# yum update --disableplugin= aliases
```

禁用 aliases 和 lang 相关插件示例如下：

```
# # yum update --disableplugin= aliases
```

2.6.2. 安装附加 yum 插件

安装附加插件通常以 yum-plugin-plugin_name 规则命令，例如 kabi 的插件包名为 kabi-yum-plugins，安装插件软件包与安装 rpm 类似。Aliases 插件安装示例如下：

```
# yum install yum-plugin-aliases
```

2.6.3. 管理 yum 插件

下述是一些常用的 yum 插件管理命令：

search-disabled-repos(subscription-manager)

search-disabled-repos 插件允许临时或永久的启用软件库用于解决软件一览。启用该插件后，当 yum 安装时因为缺少依赖包失败是，插件可以提供临时的检索软件库并重试安装，之后用户可以自行配置该软件库是否永久启用。

product-id(subscription-manager)

product-id 插件可以管理产品标示认证。product-id 是默认安装的。

langpacks(yum-langpacks)

langpacks 插件用于检索软件包安装语言，langpacks 是默认安装的。

aliases(yum-plugin-aliases)

aliases 插件用于为 yum 命令添加 aliases 命令行选项。

yum-changelog(yum-plugin-changelog)

yum-changelog 插件用于提供变更的 changelog，通过参数--changelog 命令行

添加。

yum-tmprepo(yum-plugin-tmprepo)

yum-tmprepo 插件用于检查确认软件仓库安全性，默认情况下该插件不允许禁用 gpg 检查。

yum-verify(yum-plugin-verify)

yum-verify 插件用于添加 verify, verify-rpm 以及 verify-all 命令行，以便提供系统的认证信息。

第三章 基础服务

该章节介绍如何配置系统服务、后台 daemon 以及远程访问 Kylin Linux Advanced Server V7.0 系统。

3.1. 使用 systemd 管理系统服务

3.1.1. Systemd 介绍

systemd 是 Linux 下一个与 SysV 和 LSB 初始化脚本兼容的系统和服务管理器。systemd 使用 socket 和 D-Bus 来开启服务，提供基于守护进程的按需启动策略，保留了 Linux cgroups 的进程追踪功能，支持快照和系统状态恢复，维护挂载和自挂载点，实现了各服务间基于从属关系的一个更为精细的逻辑控制，拥有前卫的并行性能。systemd 无需经过任何修改便可以替代 sysvinit。

systemd 开启和监督整个系统是基于 unit 的概念。unit 是由一个与配置文件对应的名字和类型组成的(例如：avahi.service unit 有一个具有相同名字的配置文件，是守护进程 Avahi 的一个封装单元)。unit 有以下几种类型：

表格 3-1 unit 类型介绍

Unit 类型	文件后缀	描述
Service unit	.service	守护进程的启动、停止、重启和重载是此类型
Target unit	.target	此类 unit 为其他 unit 进行逻辑分组。
Automount unit	.automount	此类 unit 封装系统结构层次中的一个自挂载点。
Device unit	.device	此类 unit 封装一个存在于 Linux 设备树中的设备。
Mount unit	.mount	此类 unit 封装系统结构层次中的一个挂载点。
Path unit	.path	此类 unit 为文件系统中的一个文件或者目录。

Scope unit	.scope	此类 unit 为外部创建进行。
Slice unit	.slice	此类 unit 为一组管理系统进程的分层 unit。
Snapshot unit	.snapshot	与 target unit 相似，快照本身不做什么，唯一的目的就是引用其他 unit
Socket unit	.socket	此类 unit 封装系统和互联网中的一个 socket。
Swap unit	.swap	此类 unit 封装 swap 设备或者 swap 文件。
Timer unit	.timer	此类 unit 封装系统时间

Systemd unit 的文件目录说明如下：

表格 3-2 Systemd unit 介绍

目录	描述
/usr/lib/systemd/system/	RPM 包安装时发布的 Systemd units。
/run/systemd/system/	运行时创建的 Systemd units，该目录任务会覆盖安装时自带的 Systemd units。
/etc/systemd/system/	系统创建与管理的 Systemd units，该目录任务会覆盖运行时 Systemd units。

3.1.1.1. 主要特性

systemd 提供以下特性：

- 基于 socket 的并行性能：为了加速整个系统启动和并行启动更多的进程，systemd 在实际启动守护进程之前创建监听 socket ，然后传递 socket 给守护进程。在系统初始化时，首先为所有守护进程创建 socket，然后再启动所有的守护进程。如果一个服务因为需要另一个服务的支持而没有完全启动，而这个连接可能正在提供服务的队列中排队，那么这个客户端进程在这次请求中就处于阻塞状态。不过只会有这一个客户端进程会被阻塞，而且仅是在这一次请求中被阻塞。服务间的依赖关系也不再需要通过配置来实现真正的并行启动(因为一次开启了所有的 socket ，如果一个服务需要其他的服务，它显然可以连接到相应的 socket)。
- D-Bus 激活策略启动服务：通过使用总线激活策略，服务可以在接入时马上启动。同时，总线激活策略使得系统可以用微小的消耗实现 D-Bus 服务的提供者与消费者的同步开启请求。(同时开启多个服务，如果一个比总线激活策略中其他服务快就在 D-Bus 中排队其请求，直到其他管理确定自己的服务信息为止)。
- 提供守护进程的按需启动策略。
- 保留了使用 Linux cgroups 进程的追踪功能：每一个执行了的进程获得

它自己的一个 cgroup ，配置 sysytemd 使其可以存放在 cgroup 中已经经过外部配置的服务非常简单。（如使用 libcgroups utilities）。

- 支持快照和系统状态恢复：快照可以用来保存/恢复系统初始化时所有的服务和 unit 的状态。它有两种主要的使用情况：允许用户暂时进入一个像 "Emergency Shell" 的特殊状态，终止当前的服务；提供一个回到先前状态的简单方法，重新启动先前暂时终止的服务。
- 维护挂载和自挂载点：systemd 监视所有的挂载点的进出情况，也可以用来挂载或卸载挂载点。/etc/fstab 也可以作为这些挂载点的一个附加配置源。通过使用 comment=fstab 选项您甚至可以标记 /etc/fstab 条目使其成为由 systemd 控制的自挂载点。
- 现了各服务间基于依赖关系的一个精细的逻辑控制：systemd 支持服务(或 unit)间的多种依赖关系。在 unit 配置文件中使用 After/Before、Requires 和 Wants 选项可以固定 unit 激活的顺序。Requires 和 Wants 表示一个正向(强制或可选)的需求和依赖关系，Conflicts 表示一个负向的需求和依赖关系。其他选项较少用到。如果一个 unit 需要启动或关闭，systemd 就把它和它的依赖关系添加到临时执行列表，然后确认它们的相互关系是否一致(或所有 unit 的先后顺序是否含有循环)。如果答案是否的话，systemd 将尝试修复它，删除可以消除循环的无用工作。

3.1.1.2. 兼容性

- 与 SysV 初始化脚本兼容：如果可能，它会利用 LSB 和 chkconfig 的头信息内容，否则，就使用其他可用信息，如：/etc/rc.d 。这些初始化脚本仅仅作为一个附加的配置源，以减少 sysytemd 服务固有的路径数目。
- /etc/fstab 配置文件：这只是另一个配置源。通过使用 comment=fstab 选项标记 /etc/fstab 条目，使 systemd 可以控制自挂载点。
- 支持简单的模板/实例机制：例如只有一个作为示例的 getty@.service 文件，而不是为六个 getty 都准备一个配置文件。接口部分甚至可以被直接继承，也就是说，可以简单的调用 avahi-autoipd@eth0.service 服务配置 dhcpcd@eth0.service ，使得字符串 eth0 的值可以直接通过通配符匹配得到。
- 在一定程度上兼容/dev/initctl 。这个兼容性实际上是为了执行 FIFO-activated 服务。(只是简单地把原先的请求转换成为 D-Bus 请求)事实上，这也意味着旧的 Upstart 和 sysvinit 中的 shutdown、poweroff 和其他相似命令可以在 systemd 中继续使用。
- 与 utmp 和 wtmp 兼容。
- 它可以控制由它催生的每一个程序。

➢ 本地配置文件使用与.desktop 文件相近的语法：很多软件架构中都有这个简单的语法的分析器。它也可以借用已有的国际化的服务描述工具，语法都是相似的，没有必要再学习新的语法。

3.1.2. 管理系统服务

systemctl 是最主要的工具。它融合 service 和 chkconfig 的功能于一体。您可以使用它永久性或只在当前会话中启用/禁用服务。

表格 3-3 service 与 systemctl 的区别

service	systemctl	描述
service name start	systemctl start name.service	用来启动一个服务 (并不会重启现有的)
service name stop	systemctl stop name.service	用来停止一个服务 (并不会重启现有的)。
service name restart	systemctl restart name.service	用来停止并启动一个服务。
service name condrestart	systemctl try-restart name.service	重启一个正在运行的服务
service name reload	systemctl reload name.service	当支持时，重新装载配置文件而不中断等待操作。
service name status	systemctl status name.service systemctl is-enabled name.service	查询服务状态
service --status-al l	systemctl list units --type service --all	显示所有服务状态

表格 3-4 chkconfig 与 systemctl 的区别

在下次启动时或满足其他触发条件时设置服务为启用	chkconfig name on	systemctl enable name.service
在下次启动时或满足其他触发条件时设置服务为禁用	chkconfig name off	systemctl disable name.service

用来检查一个服务在当前环境下被配置为启用还是禁用。	chkconfig -- list name	systemctl status name.service systemctl is-enabled name.service
输出在各个运行级别下服务的启用和禁用情况	chkconfig -- list	systemctl list-unit-files --type service
显示 unit 前置启动服务	chkconfig -- list	systemctl list-dependencies --after
显示 unit 后导启动服务	chkconfig -- list	systemctl list-dependencies --before

3.1.2.1. 显示服务

输出激活的单元：

```
# systemctl
```

以下命令等效：

```
# systemctl list-units
```

如需输出激活服务的单元，执行以下命令：

```
# systemctl list-units --type service
```

输出加载服务的单元，执行以下命令：

```
# systemctl list-units --type service --all
```

输出所有服务的单元，执行以下命令：

```
# systemctl list-units-files --type service
```

以下为显示当前所有激活服务的命令：

```
# systemctl list-units --type service
```

显示所有已安装服务单元命令如下：

```
# systemctl list-unit-files --type service
```

3.1.2.2. 显示服务状态

显示服务状态命令：

```
# systemctl status name.service
```

表格 3-5 服务单元信息列表

文件	描述
Loaded	服务是否价值
Active	服务是否激活
Main PID	当前系统服务的 PID
Status	当前系统服务的附加信息
Process	相关进程附加信息
CGroup	相关 CGroup 附加信息

显示 gdm.service.服务状态示例：

```
# systemctl status gdm.service
```

显示 gdm.service 服务启动前置依赖服务示例：

```
# systemctl list-dependencies --after gdm.service
```

显示 gdm.service 服务启动后导服务示例：

```
# systemctl list-dependencies --before gdm.service
```

3.1.2.3. 启动服务

启动服务命令：

```
# systemctl start name.service
```

启动 httpd 服务示例：

```
# systemctl start httpd.service
```

3.1.2.4. 停止服务

停止服务命令：

```
# systemctl stop name.service
```

停止 httpd 服务示例：

```
# systemctl stop httpd.service
```

3.1.2.5. 重启服务

重启服务命令：

```
# systemctl restart name.service
```

重启 httpd 服务示例：

```
# systemctl restart httpd.service
```

仅重启正在运行的服务命令：

```
# systemctl try-restart name.service
```

重载服务命令：

```
# systemctl reload name.service
```

重载 httpd 服务示例：

```
# systemctl reload httpd.service
```

3.1.2.6. 启用服务

启用服务命令：

```
# systemctl enable name.service
```

重新启用服务命令：

```
# systemctl reenable name.service
```

启用 httpd 服务示例：

```
# systemctl enable httpd.servi ce
ln - s ' /usr/lib/systemd/system/httpd. service'
' /etc/systemd/system/multi- user. target. wants/httpd. service'
```

3.1.2.7. 禁用服务

禁用服务命令：

```
# systemctl enable name.service
```

禁用 bluetooth 服务示例：

```
# systemctl disable bluetooth.service
rm ' /etc/systemd/system/dbus- org. bluez. service'
rm ' /etc/systemd/system/bluetooth. target. wants/bluetooth. service'
```

3.1.3. 管理目标

启动级别（runlevel）是一个旧的概念。现在，systemd 引入了一个和启动级别功能相似又不同的概念——目标（target）。不像数字表示的启动级别，每个目

标都有名字和独特的功能，并且能同时启用多个。一些目标继承其他目标的服务，并启动新服务。systemd 提供了一些模仿 sysvinit 启动级别的目标，仍可以使用旧的 telinit 启动级别命令切换。

启动级别 0、1、3、5、6 都被赋予特定用途，并且都对应一个 systemd 的目标。要实现用户自定义启动级别功能，可以以原有的启动级别为基础，创建一个新的目标/etc/systemd/system/<新目标>（可以参考 /usr/lib/systemd/system/graphical.target），创建 /etc/systemd/system/<新目标>.wants 目录，向其中加入额外服务的链接（指向 /usr/lib/systemd/system/ 中的单元文件）。

表格 3-6 SysV 启动级别与 systemd 目标对比

运行级别	目标单元	描述
0	runlevel0.target, poweroff.target	中断系统（halt）
1	runlevel1.target, rescue.target	单用户模式
2	runlevel2.target,	用户自定义启动级别，通常识别为级别 3。
3	runlevel3.target, multi-user.target	多用户，无图形界面。用户可以通过终端或网络登录。
4	runlevel4.target, multi-user.target	用户自定义启动级别，通常识别为级别 3。
5	runlevel5.target, graphical.target	多用户，图形界面。继承级别 3 的服务，并启动图形界面服务。
6	runlevel6.target, reboot.target	重启
emergency	emergency.target	急救模式（Emergency shell）

表格 3-7 SysV 初始化命令与 systemctl 对比

旧命令	新命令	描述
run level	systemctl list-units --type target	显示当前目标
telini trunlevel	systemctl isolate name.target	变更当前目标

3.1.3.1. 查看默认目标

查看默认目标命令：

```
# systemctl get-default
```

以下为查看默认目标单元的示例：

```
# systemctl get-default
graphical. target
```

3.1.3.2. 查看当前目标

查看当前目标命令：

```
# systemctl list-units --type target
```

输出加载目标的单元，执行以下命令：

```
# systemctl list-units --type target --all
```

以下为显示当前所有激活目标的示例：

```
# systemctl list-units  --type target
```

3.1.3.3. 变更默认目标

变更默认目标命令：

```
# systemctl set-default name.target
```

以下变更多用户目标示例：

```
# systemctl set-default multi-user.target
rm ' /etc/systemd/system/default. target'
ln - s ' /usr/lib/systemd/system/multi-user. target'
' /etc/systemd/system/default. target'
```

3.1.3.4. 变更当前目标

变更当前目标命令：

```
# systemctl isolate name.target
```

以下变更多用户目标示例：

```
# systemctl isolate multi-user.target
```

3.1.3.5. 切换救援模式

systemd.unit=rescue.target 是一个设置基本系统和救援 shell 的特殊 target unit (与运行级 1 相似)；

进入救援模式命令：

```
# systemctl rescue
```

如果需要进入救援模式且不输出日志，输以下命令：

```
# systemctl --no-wall rescue
```

切换救援模式示例：

```
# systemctl rescue
```

3.1.3.6. 切换紧急模式

systemd.unit=emergency.target 与传递保留参数的 init=/bin/sh 给系统使系统从该状态启动相似。

进入紧急模式命令：

```
# systemctl emergency
```

如果需要进入紧急模式且不输出日志，输以下命令：

```
# systemctl --no-wall emergency
```

3.1.4. 在远程机器上使用 systemd

连接远程机器使用 systemd 命令如下：

```
# systemctl --host user_name@host_name command
```

远程管理命令举例：

```
[root@localhost ~]# systemctl -H root@server-01.example.com status httpd.service
```

3.1.5. 创建和修改 systemd 单元文件

3.1.5.1. 单元文件介绍

单元文件通常包括三个部分：

【Unit】：通用配置项，包括该 unit 的基本信息。

【Unit type】：Unit 类型，不同类型定义可以参考 systemd 简介内容。

【Install】：包括需要被安装、启用和禁用的服务内容。

表格 3-8 【Unit】字段配置项说明

配置项	描述
Description	一些描述，显示给用户界面看的，可以是任何字符串，一般是关于服务的说明。
Documentation	指定参考文档的列表，以空格分开的 URI 形式，如 http://, https://, file:, info:,man，这是有顺序的，最好是先解释这个服务的目的是什么，然后是它是如何配置的，再然后是其它文件，这个选项可以多次

	指定，会将多行的合并，如果指定了一个空的，那么会重置此项，前的配置不在起作用。
After/Before	配置服务间的启动顺序，比如一个 foo.service 包含了一行 Before=bar.service，那么当他们同时启动时，bar.service 会等待 foo.service 启动完成后才启动。注意这个设置和 Requires= 的相互独立的，同时包含 After= 和 Requires= 也是常见的。此选项可以指定一次以上，这时是按顺序全部启动。
Requires	指定此服务依赖的其它服务，如果本服务被激活，那么 Requires 后面的服务也会被激活，反之，如果 Requires 后面的服务被停止或无法启动，则本服务也会停止。这个选项可以指定多次，那么就要求所有指定的服务都被激活。需要注意的是这个选项不影响启动或停止的顺序，启动顺序使用单句的 After= 和 Before= 来配置。例如，如果 foo.service 依赖 bar.serivce，但是只配置了 Requires= 而没有 After= 或 Before=，那么 foo.service 启动时会同时激活 foo.service 和 bar.service。通常使用 Wants= 代替 Requires= 是更好的选择，因为系统会更好的处理服务失败的情况。
Wants	相对弱化的 Requires= ，这里列出的服务会被启动，但如果无法启动或无法添加到事务处理，并不影响本服务做为一个整体的启动。这是推荐的两个服务关联的方式。这种依赖也可以配置文件外，通过 .wants/ 目录添加，具体可以看上面的说明。
Conflicts	配置一个依赖冲突，如果配置了些项，那么，当一个服务启动时，或停止此处列出的服务，反过来，如果这里列出的服务启动，那么本服务就会停止，即后启动的才起作用。注意，此设置和 After= 和 Before= 是互相独立的。如果服务 A 和 B 冲突，且在 B 启动的时候同时启动，那么有可能会启动失败（两都是必需的）或修改以修复它（两者之一或两都不是必需的），后一种情况，会将不需要的依赖删除，或停止冲突。

表格 3-9【Service】字段配置项

配置项	描述
Type	设置进程的启动类型，必须是下列值之一： simple, forking, oneshot, dbus, notify, idle ➢ 如果设为"simple"(设置了 ExecStart= 但未设置 BusName=

	时的默认值)，那么表示 ExecStart= 所设定的进程就是该服务的主进程。 ➢ 如果此进程需要为其他进程提供服务，那么必须在该进程启动之前先建立好通信渠道(例如套接字)，以加快后继单元的启动速度。 ➢ "dbus"(设置了 ExecStart= 与 BusName= 时的默认值)与"simple"类似，不同之处在于该进程需要在 D-Bus 上获得一个由 BusName= 指定的名称。systemd 将会在启动后继单元之前，首先确保该进程已经成功的获取了指定的 D-Bus 名称。设置为此类型相当于隐含的依赖于 dbus.socket 单元。 ➢ "oneshot"(未设置 ExecStart= 时的默认值)与"simple"类似，不同之处在于该进程必须在 systemd 启动后继单元之前退出。此种类型通常需要设置 RemainAfterExit= 选项。 ➢ 如果设为"forking"，那么表示 ExecStart= 所设定的进程将会在启动过程中使用 fork() 系统调用。这是传统 UNIX 守护进程的经典做法。也就是当所有的通信渠道都已建好、启动亦已成功之后，父进程将会退出，而子进程将作为该服务的主进程继续运行。对于此种进程，建议同时设置 PIDFile= 选项，以帮助 systemd 准确定位该服务的主进程，进而加快后继单元的启动速度。 ➢ "notify"与"simple"类似，不同之处在于该进程将会在启动完成之后通过 sd_notify(3) 之类的接口发送一个通知消息。systemd 将会在启动后继单元之前，首先确保该进程已经成功的发送了这个消息。如果设置为此类型，那么 NotifyAccess= 将只能设置为"all"或者"main"(默认)。注意，目前 Type=notify 尚不能在 PrivateNetwork=yes 的情况下正常工作。 ➢ "idle"与"simple"类似，不同之处在于该进程将会被延迟到所有的操作都完成之后再执行。这样可以避免控制台上的状态信息与 shell 脚本的输出混杂在一起。
ExecStart	在启动该服务时需要执行的命令行(命令+参数)。 仅在设置了 Type=oneshot 的情况下，才可以设置任意个命令行(包括零个)，否则必须且只能设置一个命令行。 多个命令行既可以在同一个 ExecStart= 中设置，也可以通过设置多个 ExecStart= 来达到相同的效果。

	如果设为一个空字符串，那么先前设置的所有命令行都将被清空。 如果不设置任何 ExecStart= 指令，那么必须确保设置了 RemainAfterExit=yes 指令。 命令行必须以一个绝对路径表示的可执行文件开始，并且其后的那些参数将依次作为"argv[1] argv[2] ..."传递给被执行的进程。 如果在绝对路径前加上可选的"@"前缀，那么其后的那些参数将依次作为"argv[0] argv[1] argv[2] ..."传递给被执行的进程。 如果在绝对路径前加上可选的"-"前缀，那么即使该进程以失败状态(例如非零的返回值或者出现异常)退出，也会被视为成功退出。可以同时使用"-"与"@"前缀，且顺序任意。 如果设置了多个命令行，那么这些命令行将以其在单元文件中出现的顺序依次执行。 如果某个无"-"前缀的命令行执行失败，那么剩余的命令行将不会被执行，同时该单元将变为失败(failed)状态。 当未设置 Type=forking 时，这里设置的命令行所启动的进程将被视为该服务的主守护进程。
ExecStop	这是一个可选的指令，用于设置当该服务被要求停止时所执行的命令行。语法规则与 ExecStart= 完全相同。 执行完此处设置的命令行之后，该服务所有剩余的进程将会根据 KillMode= 的设置被杀死(参见 systemd.kill(5) 手册)。 如果未设置此选项，那么当此服务被停止时，该服务的所有进程都将会根据 KillMode= 的设置被立即全部杀死。 与 ExecReload= 一样，也有一个特殊的环境变量 $MAINPID 可以用于表示主进程的 PID 一般来说，仅仅设置一个结束服务的命令，而不等待其完成，是不够的。 因为当此处设置的命令执行完之后，剩余的进程会被 SIGKILL 信号立即杀死，这可能会导致数据丢失。 因此，这里设置的命令必须是同步操作，而不能是异步操作。
ExecReload	这是一个可选的指令，用于设置当该服务被要求重新载入配置时所

	执行的命令行。语法规则与 ExecStart= 完全相同。 另外，还有一个特殊的环境变量 $MAINPID 可以用于表示主进程的 PID，例如可以这样使用： /bin/kill -HUP $MAINPID 注意，像上例那样，通过向守护进程发送复位信号，强制其重新加载配置文件，并不是一个好习惯。 因为这是一个异步操作，所以不适用于需要按照特定顺序重新加载配置文件的服务。 我们强烈建议将 ExecReload= 设置为一个能够确保重新加载配置文件的操作同步完成的命令行。
Restart	当服务进程正常退出、异常退出、被杀死、超时的时候，是否重新启动该服务。 "服务进程"是指 ExecStart=, ExecStartPre=, ExecStartPost=, ExecStop=, ExecStopPost=, ExecReload= 中设置的进程。 当进程是由于 systemd 的正常操作(例如 systemctl stop|restart)而被停止时，该服务不会被重新启动。 "超时"可以是看门狗的"keep-alive ping"超时，也可以是 systemctl start|reload|stop 操作超时。 该选项可以取下列值之一：no, on-success, on-failure, on-abnormal, on-watchdog, on-abort, always "no"(默认值)表示不会被重启。"always"表示会被无条件的重启。 "on-success"表示仅在服务进程正常退出时重启，所谓"正常退出"是指： 退出码为"0"，或者进程收到 SIGHUP, SIGINT, SIGTERM, SIGPIPE 信号并且退出码符合 SuccessExitStatus= 的设置。 "on-failure"表示仅在服务进程异常退出时重启，所谓"异常退出"是指： 退出码不为"0"，或者进程被强制杀死(包括"core dump"以及收到 SIGHUP, SIGINT, SIGTERM, SIGPIPE 之外的其他信号)， 或者进程由于看门狗或者 systemd 的操作超时而被杀死。
Remai nAfterExi	可设为"yes"或"no"(默认值)，表示当该服务的所有进程全部退出之

t	后，是否依然将此服务视为活动(active)状态。

表格 3-10 【install】字段配置项

配置项	描述
Alias	在安装使用应该使用的额外名字（即别名）。名字必须和服务本身有同样的后缀（即同样的类型）。这个选项可以指定多次，所有的名字都起作用，当执行 systemctl enable 命令时，会建立相当的链接。
RequiredBy WantedBy	在 .wants/ 或 .requires/ 子目录中为服务建立相应的链接。这样做的效果是当列表中的服务启动，本服务也会启动。
Also	当此服务安装时同时需要安装的附加服务。 如果用户请求安装的服务中配置了此项，则 systemctl enable 命令执行时会自动安装本项所指定的服务。
DefaultInstance	表示 unit 启用时默认的实例

下面是 postfix.service 单元文件的示例：

```
[Unit]
Description=Postfix Mail Transport Agent
After=syslog. target network. target
Conflicts=sendmail. service exim. service
[Service]
Type=forking
PIDFile=/var/spool/postfix/pid/master. pid
EnvironmentFile=- /etc/sysconfig/network
ExecStartPre=- /usr/libexec/postfix/aliasesdb
ExecStartPre=- /usr/libexec/postfix/chroot- update
ExecStart=/usr/sbin/postfix start
ExecReload=/usr/sbin/postfix reload
ExecStop=/usr/sbin/postfix stop
[Install]
WantedBy=multi-user. target
```

这个示例中，【Unit】字段表述服务名称与依赖冲突信息。【Service】包括基本的服务信息。Envi ronmentFile 表述预定义的该服务的环境变量，**P IDFile** 表示服务使用静态的 PID。最后，【Install】字段显示依赖该服务的内容。

3.1.5.2. 创建自定义单元文件

创建自定义单元文件的步骤：

1) 准备自定义服务的执行文件。

可执行文件可以是脚本，也可以是软件提供者的的程序，如果需要，为自定义服务的主进程准备一个 PID 文件，一保证 PID 保持不变。另外还可能需要的配置环境变量的脚本，确保所以脚本都有可执行属性并且不需要交互。

2) 在/etc/systemd/system/目录创建单元文件，并且保证只能被 root 用户编辑：

```
# touch /etc/systemd/system/name. service
# chmod 664 /etc/systemd/system/name. service
```

文件不需要执行权限。

3) 打开 name.service 文件，添加服务配置，各种变量如何配置视所添加的服务类型而定，下面是一个依赖网络服务的配置例子：

```
[Unit]
Description=service_description
After=network. target
[Service]
ExecStart=path_to_executable
Type=forking
PIDFile=path_to_pidfile
[Install]
WantedBy=default. target
```

4) 通知 systemd 有个新服务添加：

```
# systemctl daemon-reload
# systemctl start name.service
```

创建 emacs.service 文件的例子：

1) 创建文件，并确保正确权限：

```
# touch /etc/systemd/system/emacs.service
# chmod 664 /etc/systemd/system/emacs.service
```

2) 添加配置信息：

```
[Unit]
Description=Emacs:theextensible,self-documentingtexteditor
[Service]
Type=forking
ExecStart=/usr/bin/emacs--daemon
ExecStop=/usr/bin/emacsclient--eval"(kill-emacs)"
Environment=SSH_AUTH_SOCK=%t/keyring/ssh
Restart=always
[Install]
WantedBy=default.target
```

3) 通知 systemd 并开启服务：

```
#systemctl daemon-reload
#systemctl start emas.service
```

创建 sshd-second 服务的例子：

1) 拷贝 sshd_config 文件

```
#cp /etc/ssh/sshd {,-second}_config
```

2) 编辑 sshd-second_config 文件，添加 22220 的端口，和 PID 文件：

```
Port 22220
PidFile /var/run/sshd-second.pid
```

如果还需要修改其他参数，请阅读帮助。

3) 拷贝单元文件：

```
#cp /usr/lib/systemd/system/sshd{,-second}.service
```

4) 编辑单元文件 sshd-second.service

修改描述字段

```
Description=OpenSSH server daemon
```

添加 sshd.service 服务在 After 关键字之后：

```
After=syslog.target network.target auditd.service sshd.service
```

移除 sshdkey 创建：

```
ExecStartPre=/usr/sbin/sshd-keygen
```

移除这一行

在执行脚本里，添加第二 sshd 服务的配置文件：

```
ExecStart=/usr/sbin/sshd-D-f/etc/ssh/sshd-second_config $OPTIONS
```

修改后的 sshd-second.service 文件内容如下：

```
ExecStart=/usr/sbin/sshd-D-f/etc/ssh/sshd-second_config $OPTIONS
```

5) 如果使用 SELinux，添加 tcp 端口，负责 sshd-second 服务的端口就会被拒绝绑定：

```
#semanage port -a -t ssh_port_t -p tcp 22220
```

6) 设置开机启动并测试：

```
#systemctl enable sshd-second.service
#ssh -p 22220 user@server
```

确保防火墙端口也开放。

3.1.5.3. 修改已经存在的单元文件

systemd 单元配置文件默认保存在/usr/lib/systemd/system/目录，系统管理员不建议直接修改这个目录下的文件，自定义的文件在/etc/systemd/system/目录下，如果有扩展的需求，可以使用以下方案：

创建一个目录/etc/systemd/system/unit.d/，这个是最推荐的一种方式，可以参考初始的单元文件，通过附件配置文件来扩展默认的配置，对默认单元文件的升级会被自动升级和应用。

从/usr/lib/systemd/system/拷贝一份原始配置文件到/etc/systemd/system/，然后修改。复制的版本会覆盖原始配置，这种方式不能增加附件的配置包，用于不需要附加功能的场景。

如果需要恢复到默认的配置文件，只需要删除/etc/systemd/system/下的配置文件就可以了，不需要重启机器，使用如下命令应用改变就可以：

```
# systemctl restart name.service
```

扩展默认单元配置文件

为了扩展默认的单元文件配置，需要先在/etc/systemd/system/下创建一个目录，用 root 执行类似下面的命令：

```
# mkdir /etc/systemd/system/name.service.d
```

在刚才创建的目录之下创建配置文件，必须以.conf 文件结尾。

例如创建一个自定义的依赖文件，内容如下：

```
[Unit]
Requires=new_dependency
After=new_dependency
```

另外一个例子，可以配置重启的时候，在主进程退出后 30 秒在重启，配置例子如下：

```
[Unit]
Requires=new_dependency
After=new_dependency
```

推荐每次只产生一个小文件，每个文件只聚焦完善一个功能，这样配置文件很容易被移除或者链接到其他服务对的配置目录中。

为了应用刚才的修改，使用 root 执行以下操作：

```
systemctl daemon-reload
systemctl restart name.service
```

例子：扩展 httpd.service 服务配置

为了使 httpd 服务启动的时候执行用户自定义的脚本，需要修改 httpd 的单元配置文件，执行以下几步操作，首先创建一个自定义文件的目录及自定义文件：

```
#mkdir /etc/systemd/system/httpd.service.d
#touch /etc/systemd/system/httpd.service.d/custom_script.conf
```

假设自定义文件位置在/usr/local/bin/custom.sh，将这个信息添加到 custom_script.conf 自定义脚本中：

```
[Service]
ExecStartPost=/usr/local/bin/custom.sh
```

应用更改：

```
#systemctl daemon-reload
```

```
#systemctl restart httpd.service
```

3.2. OpenSSH

SSH（Secure Shell）是一个使用客户端-服务端架构，使得两个系统之间的安全通信变得容易的协议，它能够让用户远程登录到服务端主机系统中。和其它诸如 FTP 或者 Telnet 的远程通信协议不同，SSH 加密了登录会话，致使入侵者难以通过连接获取未加密的密码。

ssh 程序被设计用于替换诸如 telnet 或者 rsh，这些比较旧的、安全性不高的、用来登录远程主机系统的终端应用程序。与其相关的一个叫做 scp 的程序，用于替换诸如 rcp 这样用来在主机之间复制文件的老程序。因为这些老旧的应用程序不会加密在客户端和服务端之间传递的密码，所以应该尽可能地避免使用它们。使用安全方法登录远程系统，能够同时降低客户端系统和远程主机系统的风险。

中标麒麟高级服务器操作系统软件包含了通用的 OpenSSH 安装包——openssh，以及 OpenSSH 服务端安装包——openssh-server 和 OpenSSH 客户端安装包——openssh-clients。注意，OpenSSH 安装包依赖于 OpenSSL 的安装包 openssl-libs，这个包安装了几个重要的加密库，使得 OpenSSH 能够提供加密通信。

3.2.1. SSH 协议

3.2.1.1. 为什么使用 SSH?

潜在的入侵者有许多可以使用的工具，能够让他们中断、拦截和重新路由网络流量，从而试图获取对一个系统的访问权限。一般来说，这些威胁可以分为以下几类：

1) 拦截两个系统之间的通信

攻击者可能位于通信双方所在网络中的某处，复制他们之间传递的任何信息。他可能会拦截并保留信息，或者修改信息后将其发送给计划中的接收者。

这种攻击通常是通过使用数据包嗅探器来进行的，数据包嗅探器是一种非常常见的网络工具，可以用来捕获网络流中的数据包，并分析其内容。

2) 冒充特定主机

攻击者的系统被配置来冒充传送的预期接收者。如果攻击者的策略成功了，用户系统将不会发现它其实是在跟一个错误的主机进行通信。

这种攻击可以通过使用一种名为“DNS 缓存投毒”的技术，或者通过所谓的“IP 欺骗”来实现。在第一种示例中，入侵者使用一台被攻破的 DNS 服务器来将客户系统指向一台恶意复制主机。在第二种示例中，入侵者发送伪造的网络数据包，让这个数据包看起来好像是从一台可信任的主机发出的。

这些技术都能够拦截可能存在的敏感信息，而且如果这些拦截是出于怀有敌

意的原因，那么结果将是灾难性的。如果使用 SSH 来进行远程登录和文件复制，那么就能够极大地减少这些安全威胁。这是因为 SSH 客户端和服务端使用数字签名来验证身份。此外，客户端和服务端系统之间的所有通信都是加密的。不管尝试在通信的哪一方进行身份欺骗都是行不通的，因为每一个数据包都是使用一个只有本地系统和远程系统才知道的密钥来加密的。

3.2.1.2. 主要特性

SSH 协议提供了以下保护措施：

1) 无法伪装预期的服务端

在初始连接之后，客户端能够验证它当前所连接的服务端的确是它之前所连接过的同一个服务端。

2) 无法捕获认证信息

客户端使用强 128 位加密算法来将它的认证信息传递给服务端。

3) 无法拦截通信

在一个会话中所有发送和接收的数据都是使用 128 位加密算法加密的，使得拦截到的传输内容要解密阅读是极度困难的。

此外，SSH 协议还提供了以下选项：

1) 提供了在网络上使用图形化应用程序的安全手段

通过使用名为“X11 转发”的技术，客户端能够从服务端转发 X11（X 窗口系统）应用程序。

2) 提供了一种保护其它不安全协议的方式

SSH 协议对它发送和接收的一切进行加密。通过使用名为“端口转发”的技术，SSH 服务端可以成为一个保护其它不安全协议（例如 POP）的导管，从而增加整体系统和数据的安全性。

3) 可以用来创建安全通道

OpenSSH 服务端和客户端可以被配置来为服务端和客户端机器之间的网络流，创建一个类似于虚拟专用网络的隧道。

4) 支持 Kerberos 认证

OpenSSH 服务端和客户端可以被配置为使用 Kerberos 网络认证协议的 GSSAPI（通用安全服务应用程序编程接口）实现来进行认证。

3.2.1.3. 协议版本

SSH 目前存在两个版本：版本 1 和较新的版本 2。中标麒麟高级服务器操作系统软件中的 OpenSSH 套件使用 SSH 版本 2，该版本具有增强的密钥交换算法，不易受到版本 1 中已知漏洞的攻击。然而，为了兼容性的原因，OpenSSH 套件同样也支持版本 1 的连接。

重要说明：

为了确保您的连接的最佳安全性，建议您只要可能就使用只兼容 SSH 版本 2 的服务端和客户端。

3.2.2. SSH 连接的事件序列

以下一系列事件可以保护两个主机之间的 SSH 通信的完整性。

1) 进行加密握手，以便客户端能够验证它正在跟正确的服务端进行通信。
2) 采用对称加密算法对客户端和远程主机之间的连接的传输层进行加密。
3) 客户端向服务端进行自我认证。
4) 客户端通过加密连接和远程主机进行交互。

3.2.2.1. 传输层

传输层的主要角色是确保两个主机之间的通信是安全可靠的，包括在认证的时候以及在随后的通信期间。传输层通过对数据进行加密和解密处理，以及通过提供对数据包发送和接收时的完整性保护，来实现这一目标。传输层也提供压缩、加速信息传输的功能。

SSH 客户端联系服务端时，将进行密钥交换，使得两个系统之间能够正确地构建传输层。密钥交换时的步骤如下：

1) 交换密钥
2) 确定公钥加密算法
3) 确定对称加密算法
4) 确定消息认证算法
5) 确定哈希算法

在密钥交换期间，服务端用一个唯一的主机密钥向客户端表明自己的身份。如果客户端以前从未和这个特定的服务端通信过，服务端的主机密钥对于客户端来说是未知的，客户端将不会连接。OpenSSH 通过接受服务端的主机密钥来规避这个问题。用户知晓，并且新的主机密钥已经被接受和验证之后，继续后续过程。在之后的连接中，客户端会用已保存的版本来检查服务端的主机密钥，以确保客户端确实是在和预期的服务端通信。如果在将来主机密钥发生了变化，用户必须在连接之前删除客户端已经保存的版本。

重要说明：

攻击者可能会在最初的联系阶段伪装成 SSH 服务端，因为本地系统并不知道预期的服务端和攻击者设置的假服务端之间有什么区别。为了防止这样的事情发生，请在第一次连接或者主机密钥不匹配时，联系服务端管理员以验证 SSH 服务端的真实完整性。

SSH 被设计成几乎可以和任何类型的公钥算法或者编码格式兼容。在最初

的密钥交换创建了一个用于交换的哈希值和一个共享的密值后，两个系统立即开始计算新的密钥和算法，以保护将在连接上发送的认证和数据。

在使用指定的密钥和算法传输一定量（准确的量取决于 SSH 实现）的数据之后，将会发生又一次密钥交换，生成另一套哈希值和一个新的共享密值。即使攻击者能够确定哈希值和共享密值，这一信息也仅在非常有限的一段时间内有用。

3.2.2.2. 认证

一旦传输层构建好一个安全隧道在两个系统之间传递信息，服务端告知客户端其所支持的不同的认证方法，例如使用一个私钥编码的签名，或者输入一个密码。然后客户端尝试使用所支持的其中一种方法向服务端进行认证。

SSH 服务端和客户端可以配置使用不同类型的认证方式，让各方都具有最佳的控制度。服务端可以决定基于其安全模型，它可以支持哪些加密方法；客户端可以从可用的选项中选择尝试认证方法的顺序。

3.2.2.3. 通道

在 SSH 传输层完成一次成功的认证之后，将会通过被称为“多路技术”（多路复用连接由多个信号组成，这些信号通过一个共享的、通用的介质进行发送。对 SSH 来说，不同的通道都在一个共同的安全连接中发送）的一种技术打开多重通道。每一条通道负责处理不同的终端会话和转发 X11 会话。

不论是客户端还是服务端都可以创建新的通道。每一个通道将在连接的两端被赋予一个编号。当客户端尝试打开一个新的通道时，客户端把请求和通道编号一起发送出去。这些信息被服务端保存起来，用于将通信导向对应的通道。如此一来，不同类型的会话不会相互影响，并且当一个指定的会话结束之后，关闭它的通道不会中断主要的 SSH 连接。

通道也可以支持流控制，可以让通道以一种有秩序的方式发送和接收数据。以这种方式，只有当客户端接收到通道已经打开的消息，数据才会通过通道发送。

客户端和服务端自动协商每条通道的特征，取决于客户端请求的服务类型以及用户连接到网络的方式。这样可以在不必改变协议的基础设施的情况下，为处理不同类型的远程连接带来很大的灵活性。

3.2.3. 配置 OpenSSH

3.2.3.1. 配置文件

配置文件有两个不同的系列，一系列用于客户端程序（例如 ssh、scp 和 sftp），另外一系列用于服务端（sshd 守护进程）。

系统范围的 SSH 配置信息保存在/etc/ssh/目录下，详见表格 3-12 系统范围的配置文件。特定用户的 SSH 配置信息保存在~/.ssh/目录下（该目录在特定用户

的家目录里），详见表格 3-13 特定用户的配置文件。

表格 3-12 系统范围的配置文件

文件	描述
/etc/ssh/moduli	包含了 Diffie-Hellman 密钥交换需要使用的 Diffie-Hellman 组，Diffie-Hellman 密钥交换对于构建安全的传输层是关键的。当密钥在一个 SSH 会话的最初阶段被交换的时候，一个共享的密值被创建，该密值无法由任何单独的一方所确定。这个密值随后被用来提供主机认证。
/etc/ssh/ssh_config	默认的 SSH 客户端配置文件。注意，如果~/.ssh/config 存在的话，该文件的配置将被~/.ssh/config 覆盖。
/etc/ssh/sshd_config	sshd 守护进程的配置文件。
/etc/ssh/ssh_host_ecdsa_key	sshd 守护进程所使用的 ECDSA 私钥。
/etc/ssh/ssh_host_ecdsa_key.pub	sshd 守护进程所使用的 ECDSA 公钥。
/etc/ssh/ssh_host_ed25518_key	SSH 协议版本 1 的 sshd 守护进程所使用的 RSA 私钥。
/etc/ssh/ssh_host_ed25518_key.pub	SSH 协议版本 1 的 sshd 守护进程所使用的 RSA 公钥。
/etc/ssh/ssh_host_rsa_key	SSH 协议版本 2 的 sshd 守护进程所使用的 RSA 私钥。
/etc/ssh/ssh_host_rsa_key.pub	SSH 协议版本 2 的 sshd 守护进程所使用的 RSA 公钥。
/etc/pam.d/sshd	sshd 守护进程的 PAM 配置文件。
/etc/sysconfig/sshd	sshd 服务的配置文件。

表格 3-13 特定用户的配置文件

文件	描述
~/.ssh/authorized_keys	为服务端保留一个授权的公钥列表。当客户端连接到服务端时，服务端通过检查保存在该文件中的它的签名公钥来认证客户端。

文件	描述
~/.ssh/id_ecdsa	包含用户的 ECDSA 私钥。
~/.ssh/id_ecdsa.pub	用户的 ECDSA 公钥。
~/.ssh/id_rsa	SSH 协议版本 2 的 ssh 所使用的 RSA 私钥。
~/.ssh/id_rsa.pub	SSH 协议版本 2 的 ssh 所使用的 RSA 公钥。
~/.ssh/identity	SSH 协议版本 1 的 ssh 所使用的 RSA 私钥。
~/.ssh/identity.pub	SSH 协议版本 1 的 ssh 所使用的 RSA 公钥。
~/.ssh/known_hosts	包含用户访问的 SSH 服务端的主机密钥。该文件对于确保 SSH 客户端正在连接正确的 SSH 服务端是很重要的。

获取有关 SSH 配置文件中所使用的各种指令的信息，请参考 ssh_config(5)和 sshd_config(5)手册页面。

3.2.3.2. 启动 OpenSSH 服务端

您必须安装 openssh-server 包之后才能运行 OpenSSH 服务端。

要在当前会话中启动 sshd 守护进程，请以 root 用户在 shell 命令行提示符下输入以下命令：

```
~]# systemctl start sshd. service
```

要在当前会话中停止运行 sshd 守护进程，请以 root 用户在 shell 命令行提示符下输入以下命令：

```
~]# systemctl stop sshd.service
```

如果您想在系统启动时自动启动守护进程，请以 root 用户在 shell 命令行提示符下输入以下命令：

```
~]# systemctl enable sshd.service
ln -s '/usr/lib/systemd/system/sshd.service' '/etc/systemd/system/multiu
ser.
target.wants/sshd.service'
```

3.2.3.3. 使用 SSH 进行远程连接

为了让 SSH 真正发挥作用，应该禁止使用不安全的连接协议。否则，用户的密码可能在一个会话中使用 SSH 时被保护得很好，但是却在之后使用 Telnet 登录时被捕获了。需要禁用的服务包括 telnet、rsh、rlogin 和 vsftpd。

3.2.3.4. 使用基于密钥的认证

为了更进一步的提高系统安全，可以生成 SSH 密钥对，然后强制使用基于密钥的认证，并禁用密码认证。要这样做，请在 vi 或者 nano 等文本编辑器中打开/etc/ssh/sshd_config 配置文件，并将 PasswordAuthentication 选项修改为如下内容：

```
PasswordAuthentication no
```

如果您不是在一个新的默认安装的系统中进行操作，请检查配置文件确保没有设置 PubkeyAuthentication no 选项。如果是远程连接上的，而不是使用的控制台或者带外访问，建议在禁用密码认证之前先测试一下基于密钥的登录过程。

为了能够使用 ssh、scp 或者 sftp 从一个客户端机器连接到服务端，请按照以下步骤生成一个授权密钥对。注意，这些密钥必须针对每个用户分别生成。

中标麒麟高级服务器操作系统软件 V7.0 默认使用版本 2 的 SSH 协议和 RSA 密钥。

重要说明：

如果您以 root 的身份完成了步骤，那么只有 root 用户能够使用这些密钥。

备注：

如果您要重装您的系统，又想保留之前生成的密钥对，请备份~/.ssh/目录。在重装后，将其复制回您的家目录。这一过程需要让系统上的所有用户执行，包括 root 用户。

3.2.3.5. 生成密钥对

要生成 SSH 协议版本 2 的 RSA 密钥对，请按以下步骤操作：

1) 在 shell 命令行提示符下输入以下命令生成 RSA 密钥对：

```
~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_rsa):
```

2) 按回车键确认新创建的密钥的默认路径，即~/.ssh/id_rsa。
3) 输入一个口令，并且在提示确认的时候再次输入该口令。为了安全起见，请不要使用和您登录您的账户相同的密码。
4) 默认情况下，将~/.ssh/目录的权限设置为 rwx------或者以八进制标注表示的 700。这是为了确保只有对应的用户 USER 能够查看其内容。如果有必要，可以使用以下命令来进行确认：

```
~]$ ls -ld ~/.ssh
```

```
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5) 使用以下格式的命令，将公钥复制到一台远程机器上：

```
ssh-copy-id user@hostname
```

如果公钥尚未安装的话，该命令会复制最近一次修改的~/.ssh/id*.pub 公钥。可选的，您也可以指定公钥的文件名：

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

该命令会将~/.ssh/id_rsa.pub 的内容复制到您想连接的机器的~/.ssh/authorized_keys 文件中。如果 authorized_keys 文件已经存在了，密钥将会追加到该文件的末尾。

要生成 SSH 协议版本 2 的 ECDSA 密钥对，请按以下步骤操作：

1) 在 shell 命令行提示符下输入以下命令生成 ECDSA 密钥对：

```
~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_ecdsa):
```

2) 按回车键确认新创建的密钥的默认路径，即~/.ssh/id_ecdsa。

3) 输入一个口令，并且在提示确认的时候再次输入该口令。为了安全起见，请不要使用和您登录您的账户相同的密码。

4) 默认情况下，将~/.ssh/目录的权限设置为 rwx------或者以八进制标注表示的 700。这是为了确保只有对应的用户 USER 能够查看其内容。如果有必要，可以使用以下命令来进行确认：

```
~]$ ls -ld ~/.ssh
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5) 使用以下格式的命令，将公钥复制到一台远程机器上：

```
ssh-copy-id user@hostname
```

如果公钥尚未安装的话，该命令会复制最近一次修改的~/.ssh/id*.pub 公钥。可选的，您也可以指定公钥的文件名：

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub user@hostname
```

该命令会将~/.ssh/id_ecdsa.pub 的内容复制到您想连接的机器的~/.ssh/authori

zed_keys 文件中。如果 authorized_keys 文件已经存在了，密钥将会追加到该文件的末尾。

重要说明：

私钥仅供您个人使用，绝不要把它给任何人，这一点是非常重要的。

3.2.3.6. 配置 ssh-agent

要保存您的口令，以便在您初始化一个到远程机器的连接时，不用每次都输入口令，您可以使用 ssh-agent 认证代理。如果您正在使用 GNOME，您可以配置它在您登录时提示输入您的口令，并在整个会话中记住该口令。否则，您可以为某个确定的 shell 终端保存口令。

要在您的 GNOME 会话期间保存您的口令，请按以下步骤进行操作：

1) 确保您已经安装了 openssh-askpass 包。如果没有安装，请参见 2.2.4 安装软件包。了解如何在中标麒麟高级服务器操作系统软件中安装新的包。

2) 打开终端，输入命令 gnome-session-properties 并执行。

图 3-1 Startup Applications Preferences

3) 点击右侧的【添加】按钮，然后在【命令】文本框中输入/usr/bin/ssh-add。

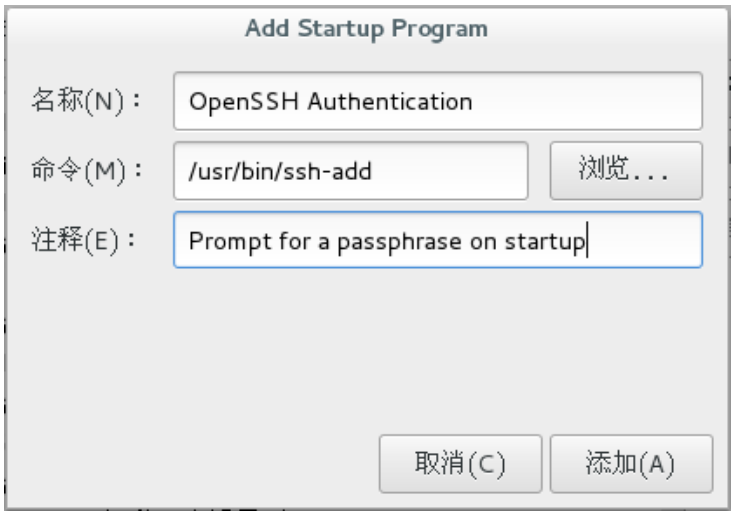


图 3-2 添加新的程序

4) 点击【添加】按钮，并确保新添加的条目旁边的复选框是选中的。

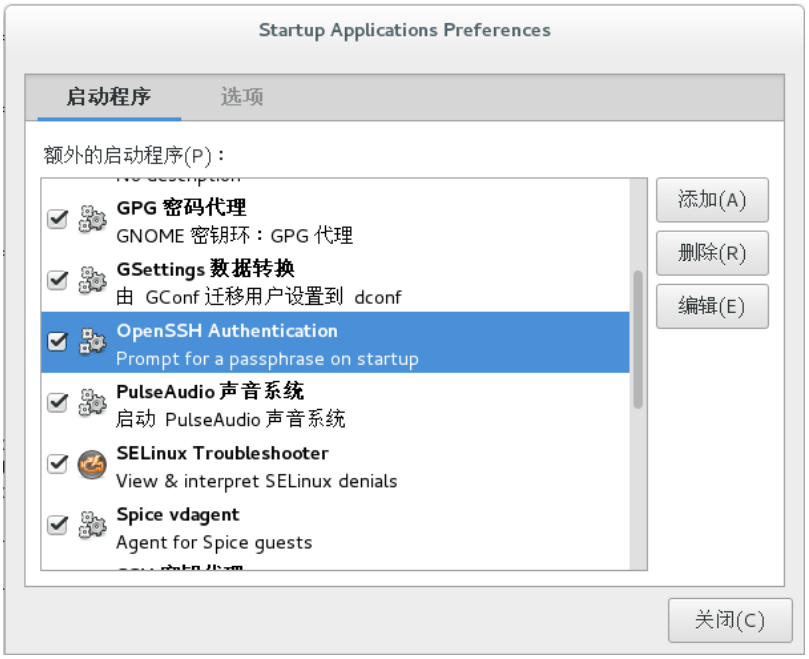


图 3-3 启用应用程序

5) 注销用户然后重新登录。将会弹出一个对话框提示您输入您的口令。从此刻起，您将不会再被 ssh、scp 或者 sftp 提示输入密码。

图 3-4 输入口令

要为某个确定的 shell 终端保存口令，可以使用以下命令：

```
~]$ ssh-add
Enter passphrase for /home/USER/.ssh/id_rsa:
```

注意，当您登出 shell 时，您的口令将被忘记。您必须在每次登录一个虚拟控制台或终端窗口时执行此命令。

3.2.3.7. OpenSSH 客户端

您必须安装 openssh-clients 包后，才能从客户端机器连接到一个 OpenSSH 服务端（请参见 2.2.4 安装软件包。了解如何在中标麒麟高级服务器操作系统软件中安装新的包）。

3.2.3.8. 使用 ssh 工具

ssh 工具可以让您登录到一台远程机器上，并在上面执行命令。它是对 rlogin、rsh 和 telnet 程序的一个安全替换。

和 telnet 命令相似，使用以下命令登录到一台远程机器上：

```
ssh hostname
```

例如，要登录到一台名为 penguin.example.com 的远程主机，可以在 shell 命令行提示符下输入以下命令：

```
~]$ ssh penguin.example.com
```

该命令将会以您正在使用的本地机器的用户名登录。如果您想指定一个不同的用户名，请使用以下命令：

```
ssh username@hostname
```

例如，以 USER 登录到 penguin.example.com，请输入以下命令：

```
~]$ ssh USER@penguin.example.com
```

您第一次初始连接时，将会看到和如下内容相似的信息：

```
The authenticity of host 'penguin.example.com' can't be established.
ECDSA key fingerprint is 256
da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff.
Are you sure you want to continue connecting (yes/no)?
```

在回答对话框中的问题之前，用户应该始终检查指印是否正确。用户可以询问服务端的管理员以确认密钥是正确的。这应该以一种安全的、事先约定好的方式进行。如果用户可以使用服务端的主机密钥，可以使用以下 ssh-keygen 命令来检查指印：

```
~]# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256 da:24:43:0b:2e:c1:3f:a1:84:13:92:01:52:b4:84:ff (ECDSA)
```

输入 yes 接受密钥并确认连接。您将会看到一个有关服务端已经被添加到已知的主机列表中的通告，以及一个输入密码的提示：

```
Warning: Permanently added 'penguin.example.com' (ECDSA) to the list of
known hosts.
USER@ penguin.example.com's password:
```

重要说明：

如果 SSH 服务端的主机密钥改变了，客户端将会通知用户连接不能继续，除非将服务端的主机密钥从~/.ssh/known_hosts 文件中删除。然而，在进行此操作之前，请联系 SSH 服务端的系统管理员，验证服务端没有收到攻击。

要从~/.ssh/known_hosts 文件中删除一个密钥，可以使用如下命令：

```
~]# ssh-keygen -R penguin.example.com
```

在输入密码之后，您将会进入远程主机的 shell 命令行提示符下。

可选地，ssh 程序可以用来在远程主机上执行一条命令，而不用登录到 shell 命令行提示符下：

```
ssh [username@]hostname command
```

例如，/etc/kylin-release 文件提供有关操作系统版本的信息。要查看 penguin.example.com 上该文件的内容，输入：

```
~]$ ssh USER@penguin.example.com cat /etc/kylin-release
USER@penguin.example.com's password:
KYLIN Linux Advanced Server V7.0 release V7.0Update2 (Potassium)
```

在您输入正确的密码之后，将会显示远程主机的操作系统版本信息，然后您将返回到本地的 shell 命令行提示符下。

3.2.3.9. 使用 scp 工具

scp 可以用来在主机之间通过一个安全加密的连接传输文件。在设计上，它和 rcp 非常相似。

要传输一个本地文件到远程系统中，可以使用如下形式的命令：

```
scp localfile username@hostname:remotefile
```

例如，如果您想将 taglist.vim 传输到名为 penguin.example.com 的远程主机上，可以在 shell 命令行提示符下输入以下命令：

```
~]$ scp taglist.vim
```

一次可以指定多个文件。要传输.vim/plugin/目录下的内容到远程主机 penguin.example.com 的相同目录下，可以输入以下命令：

```
~]$ scp .vim/plugin/* USER@penguin.example.com:.vim/plugin/
```

要将一个远程的文件传输到本地系统上，可以使用以下语法：

```
scp username@hostname:remotefile localfile
```

例如，要从远程主机上下载.vimrc 配置文件，可以输入：

```
~]$ scp USER@penguin.example.com:.vimrc .vimrc
```

3.2.3.10. 使用 sftp 工具

sftp 工具可以用来打开一个安全的、交互式的 FTP 会话。在设计上，它类似于 ftp，不同之处在于 sftp 使用了一个安全的加密连接。

要连接到远程系统中，可以使用如下形式的命令：

```
sftp username@hostname
```

例如，要使用 USER 用户登录到名为 penguin.example.com 的远程主机上，可以输入：

```
~]$ sftp USER@penguin.example.com
USER@penguin.example.com's password:
Connected to penguin.example.com.
sftp>
```

当您输入正确的密码之后，您将会看到一个 sftp 的命令行提示符。sftp 工具可以使用一系列和 ftp 类似的命令（参见表格 3-14）。

表格 3-14 常用的 sftp 命令

命令	描述
ls [directory]	列出一个远程目录的内容。如果没有提供任何目录名称，默认使用当前的工作目录。
cd directory	切换远程工作目录到指定的目录下。
mkdir directory	创建一个远程目录。
rmdir path	删除一个远程目录。
put localfile [remotefile]	将本地文件传输到远程主机上。
get remotefile [localfile]	将远程主机上的文件下载到本地主机上。

要获取可用命令的完整列表，请参考 sftp(1)用户手册页面。

3.2.4. 不只是一个安全的 Shell

一个安全的命令行界面只不过是 SSH 众多使用方式的开端。给予合适的带宽，可以通过 SSH 通道进行 X11 会话的转发。或者，通过 TCP/IP 转发，系统间以前的不安全端口连接可以映射到指定的 SSH 通道上。

3.2.4.1. X11 转发

要通过 SSH 连接打开 X11 会话，可以使用以下形式的命令：

```
ssh -Y username@hostname
```

例如，要使用 USER 用户登录到名为 penguin.example.com 的远程主机上，可以输入：

```
~]$ ssh -Y USER@penguin.example.com
USER@penguin.example.com's password:
```

当从安全 shell 命令行提示符下运行一个 X 程序时，SSH 客户端和服务端会创建一个新的安全通道，X 程序数据通过这个通道透明地发送到客户端机器上。

注意，在发生 X11 转发之前，必须在远程系统中安装好 X 窗口系统。以 root 用户输入以下命令可以安装 X11 软件包分组：

```
~]# yum group install "X Window System"
```

要了解关于软件包分组的更多信息，请参见 2.2 管理软件包。

X11 转发非常有用。例如，X11 转发可以用来为【打印设置】工具创建一个安全的交互式会话。要这样做，先使用 ssh 连接到服务端，然后输入：

```
~]$ system-config-printer &
```

【打印设置】工具将会显示出来，让远程用户可以在远程主机上安全地配置打印。

3.2.4.2. 端口转发

SSH 可以通过端口转发安全加固其他不安全的 TCP/IP 协议。在使用这一技术时，SSH 服务端成为了 SSH 客户端的一个加密导管。

端口转发的工作原理是将客户端的一个本地端口映射到服务端的一个远程端口上。SSH 可以从服务端将任意端口映射到客户端的任意端口上。这一技术并不需要端口号互相匹配。

重要说明：

要设置端口转发监听 1024 以下的端口，需要 root 级别的访问权限。

要创建一个监听 localhost 上的连接的 TCP/IP 端口转发通道，可以使用以下形式的命令：

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

例如，要使用 POP3 通过一个加密连接检查名为 mail.example.com 的服务器上的邮件，可以使用以下命令：

```
~]$ ssh -L 1100:mail.example.com:110 mail.example.com
```

一旦客户端机器和邮件服务器之间的端口转发通道准备就绪后，就可以使用一个 POP3 邮件客户端在 localhost 上使用 1100 端口来检查新邮件了。任何在客户端系统上发往 1100 端口的请求，都将被安全地导向 mail.example.com 服务器。

如果 mail.example.com 没有运行 SSH 服务端，但是同一网络中的另一台机器运行了 SSH 服务端，那么 SSH 仍然可以用来对连接进行安全加固。当然，使用的命令会略有不同：

```
~]$ ssh -L 1100:mail.example.com:110 other.example.com
```

在这一示例中，从客户端机器的 1100 端口发出的 POP3 请求，将通过 22 端口上的 SSH 连接被转发到 SSH 服务端 other.example.com。然后，other.example.com 连接到 mail.example.com 上的 110 端口来检查新邮件。注意，在使用这一技术时，只有客户端和 other.example.com 服务端之间的连接是安全的。

端口转发也可以用来通过网络防火墙安全地获取信息。如果防火墙配置为允许放行使用标准端口（即 22 端口）的 SSH 数据流，但是阻塞了对其它端口的访问，那么在要在两台主机之间使用被阻塞端口建立连接仍然是可能的，只要将它们之间的通信通过一条建立好的 SSH 连接进行重定向即可。

重要说明：

以这种方式来使用端口转发技术对连接进行转发，将允许客户端系统上的任何用户都能连接到相应的服务上。如果客户端系统被入侵，攻击者也同样能够访问转发的服务。

担心端口转发的系统管理员，可以在服务端将/etc/ssh/sshd_config 文件中的 AllowTCPForwarding 选项设置为 No，并重启 sshd 服务，来禁用端口转发功能。

3.3. TigerVNC

TigerVNC（Tiger Virtual Network Computing）是一个图形化桌面共享系统，可以让您远程控制其它计算机。

TigerVNC 采用服务端-客户端架构：服务端共享它的输出（vncserver），客户端（vncviewer）连接到客户端。

重要说明：

和以前的中标麒麟高级服务器操作系统软件发行版不一样，中标麒麟高级服务器操作系统软件 V7.0 中的 TigerVNC 使用 systemd 系统管理守护进程进行配置。配置文件/etc/sysconfig/vncserver 被替换成了/etc/systemd/system/vncserver@.service。

3.3.1. VNC 服务端

vncserver 工具用来启动一个 VNC（Virtual Network Computing）桌面。它将使用适当的选项运行 Xvnc，并在 VNC 桌面中启动一个窗口管理器。vncserver 允许用户在一台机器上并行地运行多个独立的会话，之后这些会话可以被任意数量的客户端从任意位置访问。

3.3.1.1. 安装 VNC 服务端

要安装 TigerVNC 服务端，请以 root 用户执行以下命令：

```
~]# yum install tigervnc-server
```

3.3.1.2. 配置 VNC 服务端

VNC 服务端可以配置为一个或多个用户（倘若这些用户账户存在于系统上）启动一个显示，可以配置诸如显示设置、网络地址和端口，以及安全设置等可选参数。

修改/usr/lib/systemd/system/vncserver@.service 中的 <USER> 为需要为其提供 VNC 服务的用户名称，同时，修改 PIDFile 中该用户对应的$HOME。这里，我们为 root 用户配置了一个实例，如下所示：

```
[Unit]
Description=Remote desktop service (VNC)
After=syslog.target network.target

[Service]
Type=forking

# Clean any existing files in /tmp/.X11-unix environment
ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'
ExecStart=/usr/sbin/runuser -l root -c "/usr/bin/vncserver %i"
PIDFile=/root/.vnc/%H%i.pid
ExecStop=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'

[Install]
WantedBy=multi-user.target
```

其中的%i 就是 3.3.1.3 命令中的 display_number，可以分别取值 1，2，3 等。使用 vncserver –list 命令参看可用的 VNC 服务。

3.3.1.3. 启动 VNC 服务端

要启动或者开机自启动 VNC 服务，请在命令行中指定显示编号。在 3.3.1.2 配置 VNC 服务端下的示例“为单一用户配置 VNC 显示”中使用的配置文件担任

着模板的角色，文件中的%i 将被 systemd 用显示编号替换。使用一个有效的显示编号来执行以下命令：

```
~ ]# systemctl start vncserver@:display_number.service
```

您也可以让服务在系统启动时自动启动。之后，当您登录时，vncserver 已经自动启动了。以 root 用户执行以下命令：

```
~ ]# systemctl enable vncserver@:display_number.service
```

这时候，其他用户可以使用一个 VNC 查看程序，以及定义好的显示编号和密码来连接到 VNC 服务端。假若已经安装好了一个图形化桌面，将会显示该桌面的一个实例。这个实例和目标机器上正在显示的实例是不相同的。

为两个用户和两个不同的显示配置 VNC 服务端

对于两个已经配置好的 VNC 服务端 vncserver-USER_1@.service 和 vncserver-USER_2@.service，您可以启用不同的显示编号。例如，以下命令将会使 USER_1 的 VNC 服务端在显示编号 3 上启动，而 USER_2 的 VNC 服务端将在显示编号 5 上启动：

```
~ ]# systemctl start vncserver-USER_1@:3.service
~ ]# systemctl start vncserver-USER_2@:5.service
```

3.3.1.4. 终止 VNC 会话

类似于启用 vncserver 的开机自启动，您也可以禁用该服务的开机自启动：

```
~ ]# systemctl disable vncserver@:display_number.service
```

或者，当您的系统正在运行时，您可以以 root 用户执行以下命令来停止 VNC 服务：

```
~ ]# systemctl stop vncserver@:display_number.service
```

3.3.2. 共享一个已存在的桌面

默认情况下，一个已登录的用户拥有一个由 X 服务端提供的、在显示编号 0 上的桌面。用户可以使用 TigerVNC 服务端的 x0vncserver 来共享他们的桌面。

实例：共享一个 X 桌面

要使用 x0vncserver 共享一个已登录用户的桌面，请按以下步骤操作：

1) 以 root 用户执行以下命令：

```
~ ]# yum install tigervnc-server
```

2) 为用户设置 VNC 密码：

```
~]$ vncpasswd
Password:
Verify:
```

3) 以已登录用户的身份执行以下命令：

```
~]$ x0vncserver -PasswordFile=.vnc/passwd -AlwaysShared=1
```

倘若防火墙已经配置了允许连接 5900 端口，远程查看器现在就可以连接到显示编号 0 上，并查看已登录用户的桌面了。请参见 3.3.3.2 为 VNC 配置防火墙。

3.3.3. VNC 查看器

vncviewer 是一个用来显示图形用户界面并远程控制 vncserver 的程序。

为了操作 vncviewer，有一个包含许多条目的弹出菜单，通过这些条目可以执行诸如切换到/切换出全屏模式、退出查看器等多种操作。可选地，您也可以通过终端来操作 vncviewer。在命令行中输入 vncviewer -h 可以列出 vncviewer 的参数。

3.3.3.1. 安装 VNC 查看器

要安装 TigerVNC 的客户端 vncviewer，请以 root 用户执行以下命令：

```
~ ]# yum install tigervnc
```

连接到 VNC 服务端

一旦配置好了 VNC 服务端，您就可以从任何 VNC 查看器连接到该服务端了。

3.3.3.2. 为 VNC 配置防火墙

当使用非加密连接时，firewalld 可能会阻止连接。为了让 firewalld 允许 VNC 数据包通过，您可以开放指定的 TCP 数据流端口。当使用-via 选项时，数据流通过 SSH 做了重定向，而 SSH 的端口在 firewalld 中默认是开放的。

备注：

VNC 服务端的默认端口是 5900。要得到远程桌面将被访问的端口号，可以用默认端口号加上用户分配的显示编号。例如，对于第二个显示屏：2 + 5900 = 5902。

对于显示编号 0 到 3，可以直接利用 firewalld 对 VNC 服务的支持来操作，具体是通过以下将描述的 service 选项来进行的。注意，对于大于 3 的显示编号，其对应的端口请按本节中的例子描述的方法来开放。

3.3.3.3. 使用 SSH 连接到 VNC 服务端

VNC 是一个很明显的文本网络协议，在通信中没有相应的安全机制来应对可能的攻击。为了使通信安全，您可以通过使用-via 选项来加密您的服务端-客户端连接。这将会在 VNC 服务端和客户端之间创建一个 SSH 隧道。

加密 VNC 服务端-客户端连接的命令格式如下：

```
vncviewer -via user@host:display_number
```

第四章 服务器

4.1. Web 服务器

Web 服务器能够为用户提供一种基于 web 的网络服务，通常以网页形式呈现给用户所需的网络信息。基于插件机制，现在可以浏览更多的各种文档了。因为 web 服务器使用的是超文本传输协议（HTTP），所以也称 web 服务器为 HTTP 服务器。

4.1.1. Apache HTTP 服务器

中标麒麟高级服务器操作系统软件中提供的可用 web 服务器是 Apache HTTP 服务器（httpd），httpd 的版本为 2.4，它是由 Apache 软件基金会开发的一种开源 web 服务器。

4.1.1.1. 重要变更

httpd 服务控制

随着迁移废弃了 SysV 初始化脚本，系统管理员应该使用 apachectl 和 systemctl 命令来管理服务，而不是使用 service 命令了。下面给出了查看 httpd 服务的例子：

命令

```
service httpd graceful
```

被替换成了

```
apachectl graceful
```

针对 httpd 的 systemd 单元文件和初始化脚本，有如下所示的不同操作：

- 当重新加载服务时，默认使用的是 graceful 重启方式

➢ 当停止服务时，默认使用的是 graceful 停止方式

命令

```
service httpd configtest
```

被替换成了

```
apachectl configtest
```

私用 /tmp

为了提高系统安全性，systemd 单元文件运行 httpd 守护进程时，所使用的/tmp 是私用的，和系统的/tmp 是分开的。

配置布局

当前系统中，用于模块加载的配置文件都存放在了/etc/httpd/conf.modules.d/目录下。为 httpd 提供的一些可加载附加模块的配置文件也存放于此，如：php。/etc/httpd/conf/httpd.conf 文件中的主要配置项 Include 就是用来描述/etc/httpd/conf.modules.d/目录下的 include 文件的。这就意味着在 httpd.conf 的主体之前，先要处理 conf.modules.d 下的全部配置文件。在/etc/httpd/conf.d 目录下的文件，需要由 IncludeOptional 指示项，在 httpd.conf 文件的最后加以描述，也即是说，这些文件会在 httpd.conf 主体之后被处理。

由 httpd 软件包提供的一些其它配置文件：

➢ /etc/httpd/conf.d/autoindex.conf --- 配置 mod_autoindex 目录索引

➢ /etc/httpd/conf.d/userdir.conf --- 配置可访问用户目录，例如：http://example.com/~username/ ；为了安全起见，应该禁用这种默认访问。

➢ /etc/httpd/conf.d/welcome.conf --- 在使用早期版本时，当 http://localhost/没有内容的情况下，就将其设置成了欢迎页面。

默认配置

默认情况下，提供的 httpd.conf 文件是最小配置。许多常见的配置项，如：以前默认配置中的 Timeout 或 KeepAlive 都不再被明确设置了；硬编码设置也被取代了。针对所有配置指示项的硬编码设置在手册中有详细说明。

不兼容的语法变更

如果需要从现有的 httpd 2.2 配置迁移到 httpd 2.4，则存在有大量前后不兼容的 httpd 配置语法需要修改。更多有关升级部分的 Apache 文档，请参看 http://httpd.apache.org/docs/2.4/upgrading.html 。

处理模型

在中标麒麟高级服务器操作系统软件的早期版本中，不同的多处理模型（MPM）提供了不同的 httpd 二进制文件。如：基于子进程模型可以用“prefork”代表

/usr/sbin/httpd；基于线程模型可以用“worker” 代表/usr/sbin/httpd.worker 。

在中标麒麟高级服务器操作系统软件中，只使用了单一的 httpd。3 个 MPM 作为可加载模块还是有效的：worker，profork（default）和 event，当需要加载一个 MPM 模块时，可以通过编辑/etc/httpd/conf.modules.d/00-mpm.conf 配置文件，把相关 MPM 模块前的注释符#去掉，就可以实现 MPM 模块的加载了。

包的变更

由独立的分包 mod_ldap 实现模块的 LDAP 身份验证和授权。由新的分包 mod_session 提供模块 mod_session 和 helper。由新的分包 mod_proxy_html 提供模块 mod_proxy_html 和 mod_xml2enc。这些软件包都可以从可选频道中获得。

打包文件系统布局

不再使用/var/cache/mod_proxy/目录，取而代之的是/var/cache/httpd/目录下的 proxy 和 ssl 子目录 。

httpd 软件包所提供的打包内容已经从/var/www 迁移到了/usr/share/httpd/。

- /usr/share/httpd/icons/ --- 包含了一组用于目录索引的图标。早期版本在/var/www/icons/目录下，现在迁移到了/usr/share/httpd/icons 目录下。在默认情况下，http://localhost/icons/是有效的。在/etc/httpd/conf.d/autoindex.conf 文件中，可以配置图标的位置和可用性。
- /usr/share/httpd/manual/ --- /var/www/manual/目录已经迁移到了/usr/share/httpd/manual/目录。这个目录包含了来自于 httpd-manual 软件包的内容。包含了 httpd 的 HTML 版手册。如果安装了这个软件包，则 http://lo calhost/manual /是有效的。在/etc/httpd/conf.d/manual.conf 文件中，可以配置手册的位置和可用性。
- /usr/share/httpd/error/ --- /var/www/error/目录已经迁移到了/usr/share/httpd/error/目录。在默认配置中，已删除了自定义的多国语言 HTTP 错误信息包的配置。在/usr/share/doc/httpd-VERSION/httpd-multilang-errordoc.conf 文件中，提供了配置文件样例。

身份验证，授权和访问控制

用于控制身份验证、授权和访问控制的配置指令已发生了明显的改变。在现有配置文件中使用的 Order，Deny 和 Allow 指令应适用于新的 Require 语法。有关 Apache 文档的更多详细信息，请查看 http://httpd.apache.org/docs/2.4/howto/auth.html 。

suexec

为了提高系统的安全性，已不再安装 suexec 二进制了，取而代之的是文件系统能力位集，允许更严格的权限设置。结合这一改变，suexec 也不再使用/var/log/httpd/suexec.log 日志文件了。相反，日志信息都送到了 syslog；默认情况下，

将出现在/var/log/secure 日志文件中。

模块接口

由于 httpd 改变了模块接口定义，因此，基于 httpd 2.2 构建的第 3 方模块对 httpd 2.4 是不兼容的。这样，就需要根据 httpd 2.4 模块接口定义，对那些模块代码做必要的修改，最后，再重构。有关 httpd2.4 API 变更的详细信息列表，请参看

http://httpd.apache.org/docs/2.4/developer/new_api_2_4.html 。

用于从源构建模块的 apxs 可执行文件已经从/usr/sbin/apxs 迁移到了/usr/bin/ap。

被删除模块

在中标麒麟高级服务器操作系统软件中，已经被删除的 httpd 模块列表如下：

mod_auth_mysql，mod_auth_pgsql

在 mod_authn_dbd 中，httpd 2.4 提供了内部的 SQL 数据库认证机制。

mod_perl

在 httpd 2.4 中，不再正式支持 mod_perl 了。

mod_authz_ldap

在 httpd 2.4 的分包 mod_ldap 中，用 mod_ authz_ldap 提供了对 LDAP 的支持。

4.1.1.2. 更新配置

为了更新 Apache HTTP Server version 2.2 配置文件，需要完成以下步骤：

1. 由于模块名称有可能变更了，因此，需要先确认所有模块名称是否正确。针对已经变更了名称的每个模块，需要有针对性地调整 LoadModule 指示项。
2. 在需要加载第三方模块前，需要重新编译它们。这通常意味着需要身份验证和授权模块。
3. 如果要使用 mod_userdir 模块，则在 UserDir 指示项中，需要提供目录名称（通常为 public_html）。
4. 如果要使用 Apache HTTP 安全服务器的话，则有关启用安全套接字层（SSL）协议的更多重要信息，请参看 4.1.1.8 启用 mod_ssl 模块。

可以用以下命令，检查配置文件中可能出现的错误。

```
~]# apachectl configtest
Syntax OK
```

有关如何将 Apache HTTP 服务器的配置从 2.2 版本升级 2.4 版本，请参看 h

ttp://httpd.apache.org/docs/2.4/upgrading.html 。

4.1.1.3. 运行 httpd 服务

本章描述如何启动，停止和重启 Apache HTTP，以及如何检查 Apache HTTP 的当前状态。为了能使用 httpd 服务，应先用以下命令确认是否已经安装了 httpd。

```
~]# yum install httpd
```

通常，在中标麒麟高级服务器操作系统软件中，有关一些目标概念和如何管理系统服务的更多信息，请参看 3.1 使用 systemd 管理系统服务。

启动服务

由 root 用户执行以下命令，启动 httpd 服务：

```
~]# systemctl start httpd.service
```

执行以下命令,可以使得在系统引导时，自动启动 httpd 服务：

```
# systemctl enable httpd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
```

备注：

如果要以安全服务器方式启动 Apache HTTP 服务器，则操作系统启动后，会提示需要一个用 SSL 私钥加密的密码。

停止服务

由 root 用户执行以下命令，停止 httpd 服务：

```
~]# systemctl stop httpd.service
```

执行以下命令，可以避免 httpd 服务随操作系统自启：

```
# systemctl disable httpd.service
Removed symlink /etc/systemd/system/multi-user.target.wants/httpd.service.
```

重启服务

有以下 3 种方法，可以重启 httpd 服务：

由 root 用户执行以下命令，完全重启服务

```
~]# systemctl restart httpd.service
```

这里，首先要停止这个运行着的 httpd 服务，然后再重启它。通常，在安装或删除一个动态加载模块（如：PHP）时，会用这个命令。

由 root 用户执行以下命令，可以重新加载配置：

```
~]# systemctl reload httpd.service
```

这将会导致运行着的 httpd 服务去重新加载它的配置文件。当前正在处理的任何请求都将会被中断，客户端浏览器上也会看到有错误信息提示或页面不完整。

为了重新加载配置而又不影响当前执行着的请求，可以由 root 用户执行以下命令：

```
~]# apachectl graceful
```

这将会导致运行着的 httpd 服务去重新加载它的配置文件，当前正在处理的任何请求都将会沿用旧的配置继续处理。

在中标麒麟高级服务器操作系统软件中，有关如何管理系统服务的更多信息，请参看 3.1 使用 systemd 管理系统服务。

验证服务状态

在 shell 提示符下，执行以下命令，可以检查运行着的 httpd 服务的状态：

```
~]# systemctl is-active httpd.service
active
```

4.1.1.4. 编辑配置文件

默认情况下，当启动 httpd 时，会读取表格 4 -1 httpd 服务配置文件中列出的 httpd 服务配置文件。

表格 4 -1 httpd 服务配置文件

路　径	描　述
/etc/httpd/conf/httpd.conf	主要配置文件
/etc/httpd/conf.d/	包含在主配置文件中的其它配置文件的所在目录

虽然，默认配置能适合于大多数情况的使用，然而，熟悉一些其它更重要的配置选项也是很有必要的。注意：为了使任一修改都能生效，完成修改后必须重

启 web 服务器。想了解如何重启 httpd 服务的更多信息，请参看 4.1.1.3 运行 httpd 服务中的重启服务。

可以用以下命令，检查配置文件中可能出现的错误：

```
~]# apachectl configtest
Syntax OK
```

解决错误的简单方法就是将配置文件恢复到修改前的状态。

4.1.1.5. 使用模块

作为模块化结构的应用，httpd 服务器发布时带有大量的动态共享对象（DSOs），根据需要它们可以在运行中被动态加载或卸载。在中标麒麟高级服务器操作系统软件中，这些模块被存放在/usr/lib64/httpd/modules/目录下。

加载模块

为了加载指定的 DSO 模块，需要用到 LoadModule 指示项。注意：以独立软件包形式提供的模块，通常，在/etc/httpd/conf.d/目录下都有它自己的配置文件。

实例：加载模块

```
LoadModule ssl_module modules/mod_ssl.so
```

完成后，需要重启 web 服务器来加载模块。有关如何重启 httpd 服务的更多信息，请参看 4.1.1.3 运行 httpd 服务中的重启服务。

写模块

如果要创建新的 DSO 模块，则需要安装 httpd-devel 软件包。执行以下命令，就可以完成安装。

```
~]# yum install httpd-devel
```

这个软件包包含有构建模块所需的 include 文件，header 文件和编译生成工具 Apache eXtenSion(apxs)应用程序。

一旦写好源码，就可用以下命令来构建模块了：

```
~]# apxs -i -a -c module_name.c
```

模块生成后，就可以用加载 Apache HTTP 服务器其它模块的方法来加载它。

4.1.1.6. 设置虚拟主机

可以使用 Apache HTTP 服务器的内置虚拟主机特性，实现基于不同 IP，主机名和端口号提供的不同网络信息服务。

为了设置基于主机名的虚拟主机，可以拷贝/usr/share/doc/httpd-VERSION/httpd-vhosts.conf 配置文件到/etc/httpd/conf.d/目录下，并替换掉@@Port@@和@@ServerRoot@@占位符。根据需要自定制的选项，如下图实例所示。

实例：虚拟主机配置实例

```
<VirtualHost *:80>
    ServerAdmin webmaster@penguin.example.com
    DocumentRoot "/www/docs/penguin.example.com"
    ServerName penguin.example.com
    ServerAlias www.penguin.example.com
    ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
    CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

注意 ServerName 必须是有效的 DNS 名。<VirtualHost>容器的可自定义性很好，可以接受主服务器中大多数指示项的配置。容器中包含的 User 和 Group 在此不支持了，取而代之的是 SuexecUserGroup。

备注：

如果配置的虚拟主机监听端口不是默认的，则需要确认是否根据要求，修改了/etc/httpd/conf/httpd.conf 中的全局指示项 Listen。

为了激活新配置的虚拟主机，需要重启 web 服务器。有关如何重启 httpd 服务的更多信息，请参看 4.1.1.3 运行 httpd 服务中的重启服务。

4.1.1.7. 创建 SSL 服务器

安全套接字层（SSL）是一种能使服务器和客户端进行安全数字通讯的加密协议。传输层安全（TLS）协议就是 SSL 的扩展和改进版本，能够确保隐私和数据的安全性和完整性。Apache HTTP 服务器和 mod_ssl 模块的结合，使得使用了 OpenSSL 工具包的模块，能够提供对 SSL/TLS 的支持，通常称它为 SSL 服务器。中标麒麟高级服务器操作系统软件也支持基于 TLS 实现的 Mozilla NSS。mod_nss 模块提供了对 Mozilla NSS 的支持。

不像 HTTP 连接，任何能够拦截到它的人都可以读和修改它。所谓 HTTPS 就是在 HTTP 上增加了对 SSL/TLS 的支持，它能保护传输内容不被窃听和修改。本章主要介绍在 Apache HTTP 服务器配置上，如何启用此类模块的一些基本方法，指导如何生成私钥和自签证书的过程。

概述证书和安全

密钥主要用于安全通讯。在传统或对称加密技术中，事务的两端运用相同的密钥来解码彼此的传输。然而，在公共或非对称加密技术中，有 2 个密钥共存：一个是要保密的私钥；另一个是可以共享的，公共公钥。使用公钥编码的数据只能用对应的私钥来解码；用私钥编码的数据只能用对应的公钥来解码。

为了基于 SSL 实现安全通讯，SSL 服务器必须要使用由证书颁发机构（CA）签名的数字证书。证书中包含有服务器的各种属性（例如：服务器主机名；公司名；地理位置等等），数字签名使用的是 CA 的私钥。这个签名可以确保特定证书颁发机构已签名了证书，并且不能以任何方式修改它。

当浏览器基于 web 要建立新的 SSL 连接时，会检查 web 服务器提供的证书。如果证书没有被可信 CA 机构签名，或者证书中的主机名和连上来的主机名不匹配，web 服务器会拒绝建立通讯，通常用户也会收到相应的错误信息。

默认情况下，大多数浏览器都配置有一组被广泛使用的合法签名证书。正因为如此，设置安全服务器时，总可以选择一个合适的 CA 证书，这样，最终用户就可以建立可信连接了，否则，会收到相应的错误信息，这时，需要人为接受一个证书。由于用户可以忽略证书错误，但是，却会使得攻击者可以拦截连接，因此，强力推荐尽可能使用可信 CA。相关的更多信息，请参看表表格 4-2 查看浏览器中通常可用的 CA。

表格 4-2 查看浏览器中通常可用的 CA

web 浏览器	链 接
Mozilla Firefox	Mozilla root CA 列表
Opera	Opera 可用的根证书
Internet Explorer	Microsoft Windows 可用的根证书
Chromium	Chromium project 可用的根证书

在建立 SSL 服务器时，需要生成一个证书请求和一个密钥，然后，发送证书请求，公司的身份证明和支付到证书颁发机构。一旦 CA 验证了您的证书请求和身份证明，它将会发送一个可以和服务器一起使用的签名证书给您。另外，可以创建一个自签名的证书，它不包含有 CA 的数字签名，因此，这仅仅适用于以测试为目的场合。

4.1.1.8. 启用 mod_ssl 模块

如果想基于 mod_ssl 模块，建立一个 SSL 或 HTTPS 服务器的话，则不能有另外一个应用或模块（如：mod_nss）使用相同的端口。端口 443 是默认 HTTPS 端口。

为了使用 mod_ssl 模块和 OpenSSL 工具包，建立一个 SSL 服务器，则需要安装 mod_ssl 和 openssl 软件包。由 root 用户执行以下命令就可以完成安装：

```
~]# yum install mod_ssl openssl
```

此时，会创建 mod_ssl 的配置文件/etc/httpd/conf.d/ssl.conf，默认情况下，它包含在 Apache HTTP 服务器的主配置文件中。为了加载模块，需要重启 httpd 服务，

请参看 4.1.1.3 运行 httpd 服务中的重启服务。

重要说明：

正如 POODLE: SSLv3 vulnerability (CVE-2014-3566)中所描述的，Kylin 不推荐启用 ssl，建议仅使用 TLSv1.1 或 TLSv1.2。使用 TLSv1.1 可以实现向后兼容。虽然许多产品多已支持使用 SSLv2 或 SSLv3 协议了，并默认启用了它们，但是，现在还是不适合强力推荐使用它们。

在 mod_ssl 中启用和禁用 SSL 和 TLS

为了启用和禁用指定版本的 SSL 和 TLS 协议，既可以在配置文件中，通过在“ ## SSL Global Context” 部分添加/删除 SSLProtocol 指示项来全局启用/禁用 SSL，也可以在每个“VirtualHost”的“# SSL Protocol support”中单独编辑默认项来实现。如果没有在每个域的主机部分指定它，则将会继承全局部分的设置。为了禁用一个协议版本，管理员既可以仅在“ SSL Global Context” 部分来指定 SSLProtocol，也可以在每个域的虚拟主机部分指定它，注意要带上参数 all。

4.1.1.9. 使用存在的密钥和证书

可以用已有密钥和证书来配置 SSL 服务器，而无需创建新的。但是，也有 2 种情形是不可以的：

1. 正在修改 IP 或域名

   证书是针对特定的 IP 和域名而生成的。因此，只要其中之一发生了变化，证书就会无效。

2. 正在更改由 VeriSign 颁发了证书的服务器软件

   VeriSign 是一个常用的证书颁发机构，它会针对特定的软件，IP 地址和域名颁发证书。一旦更改了软件产品，证书就会无效。

无论上述哪种情况发生，都需要重新生成证书。有关更多相关信息，请参看 4.1.1.9 生成新密钥和证书。

如果要使用已有的密钥和证书，则要分别迁移相关文件到/etc/pki/tls/private/和 /etc/pki/tls/certs/目录下。可以执行以下命令实现：

```
~]# mv key_file.key /etc/pki/tls/private/hostname.key
~]# mv certificate.crt /etc/pki/tls/certs/hostname.crt
```

然后，在/etc/httpd/conf.d/ssl.conf 配置文件中，添加下列行：

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

为了使更新的配置马上生效，请参看 4.1.1.3 运行 httpd 服务中的重启服务。

例如：使用来自 Kylin 安全 web 服务器的密钥和证书，如下所示：

```
~]# mv /etc/httpd/conf/httpsd.key /etc/pki/tls/private/kylinos.example.com.key
~]# mv /etc/httpd/conf/httpsd.crt /etc/pki/tls/certs/kylinos.example.com.crt
```

4.1.1.10.生成新密钥和证书

为了生成新密钥和证书，在系统中必须安装了 crypto-utils 软件包。由 root 用户执行以下命令可以完成安装：

```
~]# yum install crypto-utils
```

这个软件包提供了一组生成和管理 SSL 证书和密钥的工具及 genkey 应用程序，它能帮助我们生成密钥。

重要说明：

如果想用一个新证书替换掉已有的有效证书，那就只需指定一个不同的序列号。这样，可以确保客户端浏览器知晓证书更新了，避免访问页面出错。为了用自定义序列号创建新证书，需要由 root 用户用下列命令替代 genkey 命令：

```
~]# openssl req -x509 -new -set_serial number -key hostname.key -out hostname.crt
```

备注：

在系统中，如果有针对特指主机名的密钥文件存在的话，则执行 genkey 应用程序会失败。此时，就要由 root 用户执行以下命令，删除此密钥文件：

```
~]# rm /etc/pki/tls/private/hostname.key
```

由 root 用户执行 genkey 应用程序时，需要为其选择一个合适的主机名（例如：kylinos.example.com），如下所示：

```
~]# genkey kylinos.example.com
```

4.1.1.11.为 HTTP 和 HTTPS 配置防火墙

默认情况下，中标麒麟高级服务器操作系统软件是不启用 HTTP 和 HTTPS 的。把系统配置成一个 web 服务器后，利用 firewalld 服务，通过防火墙，就可以启用 HTTP 和 HTTPS 了。

由 root 用户执行以下命令，可以启用 HTTP：

```
~]# firewall-cmd --add-service http
 success
```

由 root 用户执行以下命令，可以启用 HTTPS：

```
~]# firewall-cmd --add-service https
```

```
success
```

注意系统重启后，这些修改就失效了。为了使这些修改永久生效，只需要在执行上述命令时，加上--permanent 选项。

检查 HTTP 和 HTTPS 的网络访问许可

由 root 用户执行以下命令，可以检查通过防火墙允许访问的那些配置：

```
~]# firewall-cmd –list-all
```

这是一个默认安装的例子，虽然防火墙启用了，但是，HTTP 和 HTTPS 是不允许通过防火墙被访问的。

一旦允许通过防火墙可以访问 HTTP 和 HTTPS 了，那么执行上述命令，就会看到 services 行上会有如下的信息：

```
services: dhcpv6-client http https ssh
```

4.2. 目录服务器

4.2.1. OpenLDAP

LDAP（轻量目录访问协议）是一组开放的协议，用来访问在网络上集中存储的信息。它基于 X.500 目录共享标准，但是更少的复杂度和资源密集型，所以，LDAP 被称作 X.500 标准基础上产生的一个简化版本。

像 X.500 一样，LDAP 采用目录组织分层方式。这些目录可以存储各种信息，如名字，地址，和电话号码，甚至可以像类似于网络信息服务(NIS)来使用，确保任何人在任何机器上通过 LDAP 允许的网络都可以访问他们的账户。

LDAP 通常用于集中管理用户和组，用户身份验证或系统配置。它也能作为一个虚拟电话目录，允许用户方便的访问其他用户的信息。此外，它可以引用用户到其他 LDAP 服务器中，从而提供一个特别的全球信息的存储库。然而，它是最常见的是应用于单个组织机构，例如大学、政府部门和私人公司。

4.2.1.1. LDAP 介绍

使用 client-server 体系结构，LDAP 创建一个通过网络可访问的中心信息目录。当客户端尝试修改这个目录里面的信息时，服务器端会验证用户是否有修改的权限，然后如果有需求会添加或更新入口。为了确保对话是安全的，Transport Layer Security (TLS)密码协议被用来防止通过截获传输来攻击。

LDAP 服务器支持多种数据库系统，管理员可以根据不同需求，有更多的选择。因为已经有定义好的编程接口，能够和 LDAP 服务器进行通信的应用程序有很多，并且在数量和质量上都在增加。

4.2.1.2. LDAP 术语

下面列出在本章中使用 LD AP-specific 术语：

entry

LDAP 目录的单一组件。每一个 entry 通过单独的分辨名来定义（DN）

attribute

信息是直接和 entry 关联的。例如，假如一个组织表示为一个 LDAP 的 entry，和该组织关联的 attributes 包括地址，传真机号等等。类似的，个人表示为一个 entry，包括个人电话号码或邮箱地址。

一个 attribute 可以是一个单一的值，或者是一组无序的值列表。虽然某些属性是可选的,但其他是必需的。必要的 attributes 必须使用 objectClass 来定义，并且在/etc/openldap/slapd.d/cn=config/cn=schema/目录下的 schema 文件中被找到。

该 attribute 的声明和它的值被称为 Relative Distinguished Name (RDN)，不同于 DN，RDN 对于一个 entry 来说是独一无二的。

LDIF

LDAP 数据交换格式（LDIF），是 LDAP entry 的纯文本表现方式，如下所示：

```
[id] dn: distinguished_name
attribute_type: attribute_value…
attribute_type: attribute_value…
…
```

id 选项是一个应用定义的数字，用来编辑 entry。每一个 entry 可以包含多个 attribute_type 和 attribute_value 组，只有它们在相应的文件中被定义了。在 entry 最后，包含一空白行。

4.2.1.3. OpenLDAP 特性

OpenLDAP 提供了很多重要的特性：

- 支持 LDAPv3——在 LDAP 版本 2 当中，该协议中许多修改设计是用来保证 LDAP 更加安全。其它改进，包括支持 Simple Authentication and Security Layer (SASL), Transport Layer Security (TLS)和 Secure Sockets Layer (SSL) 等协议
- LDAP 使用 IPC——使用 inter-process communication (IPC)，加强了安全性通过消除通过网络进行对话的需求。
- IPv6 的支持——OpenLDAP 支持 IPv6 网络协议

- LDIFv1 的支持——OpenLDAP 支持 LDIF 版本 1
- 更新后的 C 接口——当前的 C 接口增强了程序员连接和使用 LDAP 目录服务器的方法
- 加强了独立的 LDAP 服务器——包括了更新了的访问控制系统，线程池，更好的工具等等。

4.2.1.4. OpenLDAP 服务器安装

在 Kylin 系统上安装 LDAP 服务器步骤如下：

1) 安装 OpenLDAP 组件，参见 4.2.2 安装 OpenLDAP 组件。
2) 配置参见 4.2.3 配置 OpenLDAP 服务器 11.。
3) 启动 slapd 服务，参见 4.2.5 运行 OpenLDAP 服务。
4) 使用 ldapadd 功能添加 entries 给 LDAP 目录。
5) 使用 LDAPsearch 功能确保 slapd 服务已经正确获得信息。

4.2.2. 安装 OpenLDAP 组件

OpenLDAP 的函数库和工具由以下包提供：

表格 4-3 OpenLDAP 包列表

Package	Description
openldap	该包包含运行 OpenLDAP 服务器和客户端应用所需要的函数库
openldap-clients	该包包含了可以访问和修改 LDAP 服务器的命令行程序
openldap-servers	该包包含了运行 LDAP 服务器的服务以及配置程序，包含了单独的 LDAP Daemon，slapd
compat-openldap	该包包含了 OpenLDAP 的兼容库函数

此外，以下包通常和 LDAP 服务器使用：

表格 4-4 常用附加 LDAP 包列表

Pckage	Description
nss-pam-ldapd	该包包含 nslcd，一个本地的 LDAP 名称服务，允许用户执行本地查询
mod_ldap	该包包含 mod_authnz_ldap 和 mod_LDAP 模块，其中 mod_authnz_ldap 模块是为 Apache HTTP Server 的 LDAP 验证模块。该模块能针对 LDAP 目录够验证用户的证书，并且能够强制访问控制基于用户名，完全的 DN，组关系，一个任意 attribute，或者是一个完整的过滤字符串。这个 mod_

	LDAP 模块包含在同一个包中，提供一个可配置的共享内存 cache，为了避免重复的 HTTP 请求，支持 SSL/TLS。

使用 yum 安装以下包：

```
yum install package…
```

例如，安装 LDAP 服务器

```
~]# yum install openldap openldap-clients openldap-servers
```

注意您必须拥有超级用户权限，使用 root 用户进行登录后运行命令。如果想知道更多安装新软件包的信息，请参考“2.2.4 安装软件包”。

4.2.2.1. OpenLDAP 服务器端程序

为了执行管理任务，这些 openldap-servers 包安装了以下组件和 slapd 服务：

表格 4-5 OpenLDAP 服务端工具列表

Command	Description
slapacl	允许您检查访问属性的列表
slapadd	允许您添加 entries 从 LDIF 文件到 LDAP 目录
slapauth	允许您检查认证和权限 ID 列表
slapcat	允许您从 LDAP 目录中以 LDIF 格式保存 entries
slapdn	允许您检查 DN 列表
slapindex	允许您根据当前内容重新编排 slapd 目录。当您修改配置文件相关参数，允许该组件
slappasswd	允许您创建一个加密的用户密码，为 ldapmodify 组件，或者是用在 slapd 配置文件中
slapschema	允许您通过相应的模式检查数据库是否符合规范
slaptest	允许您检查 LDAP 服务器配置

4.2.2.2. OpenLDAP 的客户端程序

openldap-clients 安装包包含以下组件，功能包括在 LDAP 目录下添加，修改和删除 entries：

表格 4-6 OpenLDAP 客户端工具列表

命令	描述
ldapadd	允许您给 LADP 目录添加 entries，可以通过一个文件或者

	是标准输入，该命令是 ldapmodify –a 的一个链接
ldapcompare	允许您拿一个给定的 attribute 和 LDAP 目录 entry 进行比较
ladpdelete	允许您删除一个 LDAP 目录 entry
ldapexop	允许您使用 LDAP 扩展操作
ldapmodify	允许您修改 LDAP 目录 entry，通过文件或标准输入
ldapmodrdn	允许您修改修改 LDAP entry 的 RDN 值
ldappasswd	允许您修改 LDAP 用户密码
ldapsearch	允许您修改查找 LDAP entry
ldapurl	允许您生成或分解 LADP URLS
ldapwhoami	允许您在 LDAP 服务器上执行我是谁操作

除了 ldapsearch 命令外，其它命令建议使用文件的方式进行修改，而不是直接使用命令进行修改。可以通过 man 命令查看文件格式。

4.2.2.3. LDAP 客户端应用介绍

虽然有许多 LDAP 客户端可以创建和修改服务器端目录，但是中标麒麟高级服务器操作系统软件没有包含任何其中一种。常见的能够以只读模式访问服务器目录的应用，包括 Mozilla，Thunderb ird ，Evolution，或者 Ekiga。

4.2.3. 配置 OpenLDAP 服务器

默认的，OpenLDAP 配置文件保存在/etc/openldap 目录下。下面的表格包含了最重要的一些文件和目录。

表格 4-7 OpenLDAP 配置目录和文件列表

路径	描述
/etc/openldap/ldap.conf	使用 OpenLDAP 库函数的客户端应用的配置文件，包括 ldapadd，ldapsearch，Evolution 等等
/etc/openldap/slapd.d/	Slapd 的配置文件

OpenLDAP 不在使用/etc/openldap/slapd.conf 配置文件，它使用一个配置数据库在/etc/openldap/slapd.d/目录。假如您之前的安装已经有了 slapd.conf 文件，您可以使用以下命令进行转换：

```
#slaptest –f /etc/openldap/slapd.conf –F /etc/openldap/slap.d/
```

slapd 配置包括 LDIF entries，在一个分层的目录组织结构中，可以进行相应的编辑。

4.2.3.1. 修改全局配置

LDAP 服务器的全局配置文件保存在/etc/openldap/slapd.d/cn=config.ldif 文件中。常用以下指令：

olcAllows

olcAllows 命令运行您指定哪些特性是可以使用的，使用以下格式：

```
olcAllows： feature
```

可用的特性，参照表格 4-8 可用的 olcAllows 选项。默认的选项是 bind_v2。

表格 4-8 可用的 olcAllows 选项

选项	描述
bind_v2	LDAP 可接受的 bind 请求
bind_anon_cred	当 DN 是空的时候接受匿名的 bind
bind_anon_dn	当 DN 是非空的时候接受匿名 bind
update_anon	接受匿名升级操作
proxy_authz_anon	接受匿名代理控制

例如：使用 olcAllows 指令

```
olcAllows： bind_v2 update_anon
```

olcConnMaxPending

olcConnMaxPending 命令允许您指定匿名会话最大请求等待数，命令如下：

```
olcConnMaxPending：100
```

olcConnMaxPendingAuth

olcConnMaxPendingAuth 命令您指定验证过的会话最大请求等待数，命令如下：

```
olcConnMaxPendingAuth ： number
```

默认值是 1000。

例如：使用 olcConnMaxPendingAuth 命令

```
olcConnMaxPendingAuth ： 1000
```

olcDisallows

olcDisallows 命令允许您指定哪些特性不可用。命令如下：

```
olcDisallows：feature…
```

可接受的特性列表参考表格 4-8 可用的 olcAllows 选项。没有默认不可使用的特性。

表格 4-9 可用 olcDisallows 选项

选项	描述
bind_anon	不允许接收匿名 bind 请求
bind_simple	不允许简单的 bind 验证机制
tls_2_anon	不允许增强匿名会话当收到 STARTTLS 命令
tls_authc	当已经验证后不允许 STARTTLS 命令

例如：使用 olcDisallows 命令

```
olcDisallows：bind_anon
```

olcIdleTimeout

olcIdleTimeout 命令允许您指定在关闭一个 idle 连接的时候，可以等待的时间。命令如下：

```
olcIdleTimeout：number
```

该选项默认值是不使用（意思是设置为 0）

例如：使用 olcIdleTimeout 命令

```
olcIdleTimeout：180
```

olcLogFile

olcLogFile 命令指定一个文件记录日志信息，命令如下：

```
olcLogFile：file_name
```

日志信息默认使用错误标准输入格式

例如：使用 olcLogFile 命令

```
olcLogFile：/var/log/slapd.log
```

olcReferral

olcReferral 选项允许您指定一个服务器的 URL 来处理请求，命令如下：

```
olcReferral：URL
```

默认不使用

例如：使用 olcReferral 命令

```
olcReferral：ldap://root.openldap.org
```

olcWriteTimeout

olcWriteTimeout 选项允许您指定在关闭一个未完成的写请求连接的时候，可以等待的时间。格式如下：

```
olcWriteTimeout
```

默认不开启（意思是值为 0）

例如：使用 olcWriteTimeout 命令

```
olcWriteTimeout：180
```

4.2.3.2. 修改特定数据库配置

默认的，OpenLDAP 服务器使用 Berkeley DB(BDB)作为后端数据库。该数据库配置存储在/etc/openldap/slapd.d/cn=config/olcDatabase= {1}bdb.ldif 文件。以下为配置数据库命令：

olcReadOnly

olcReadOnly 命令允许您使用只读模式使用数据库，使用如下：

```
olcReadOnly：boolean
```

接收参数：TRUE（只读模式）或 FALSE（可修改模式），默认为 FALSE

例如：使用 olcReadOnly 命令

```
olcReadOnly：TRUE
```

olcRootDN

olcRootDN 命令可以为 LDAP 目录指定超级用户。使用如下：

```
olcRootDN：distinguished_name
```

接收 DN。默认值为 cn= Manager,dn=my-domain,dc=com。

例如：使用 olcRootDN 命令

```
olcRootDN：cn=root，dn=example，dn=com
```

olcRootPW

olcRootPW 命令允许您为用户设置密码，使用如下：

```
olcRootPW：password
```

可以接收没有加密的文本字符串，或者是哈希值，在 shell 终端，使用如下命令生成哈希值

```
~]$ slappaswd
New password:
Re-enter new password:
{SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

例如：使用 olcRootPW 命令

```
olcRootPW：{SSHA}WczWsyPEnMchFf1GRTweq2q7XJcvmSxD
```

olcSuffix

olcSuffix 命令允许您指定提供信息的域，使用如下：

```
olcSuffix：domain_name
```

接收 FQDN，默认是 dc=my-domain，dc=com

例如：使用 olcSuffix 命令

```
olcSuffix：dc=example，dc=com
```

4.2.3.3. 架构扩展

自从 OpenLDAP2.3 以来，/etc/openldap/slapd.d/目录包含了之前存放在/etc/openldap/schema/目录下的一些 LDAP 定义。OpenLDAP 使用这些可以扩展方案，支持额外的 attribute 类型和对象类使用缺省架构文件。关于这个方面更多的信息，可以参考 http://www.openldap.org/doc/admin/schema.html。

4.2.3.4. 建立安全连接

OpenLDAP 客户端和服务器端使用 Transport Layer Security (TLS)框架来保证安全。TLS 是提供网络通信安全的加密协议。上面提到的，在中标麒麟高级服务器操作系统软件中 OpenLDAP 使用 Mozilla NSS 作为 TLS 的安装实现。

使用 TLS 建立安全连接，怎么通过 Mozilla NSS 使用 TLS/SSL，链接 http://www.openldap.org/faq/data/cache/1514.html。因此，需要在客户端和服务器端进行相关配置。至少，服务器端需要配置 CA 证书和它本身的服务器证书和私钥。客户端需要配置包含可信任的 CA 证书文件。

典型的，服务器端需要指定一个 CA 证书，客户端想要连接到一个安全的服务器端，因此需要在其配置文件里面指定可信任的 CA。

服务器配置：

该章节列出了在 OpenLDAP 服务器中使用 TLS，需要对 slapd 命令进行全局配置，在/etc/openldap/slapd.d/cn=config.ldif 配置文件中配置。

老版本的配置使用单独的配置文件，通常是/usr/local/etc/openldap/slapd.conf，新版的使用 slapd 后端数据库保存配置，保存目录/usr/local/etc/openldap/slapd.d/。

以下命令对于确立 SSL 来说也是有效的，除了 TLS 命令外，您需要在服务器端给 SSL 打开一个端口，一般来说是 636 端口。编辑/etc/sysconfig/slapd 文件，添加 ldaps:///字符串，指定 SLAPD_URLS 命令 URLs。

olcTLSCACertificateFile

olcTLSCACertificateFile 命令指定了 Privacy-Enhanced Mail (PEM)编码的文件，包含可信的 CA 证书。使用如下

```
olcTLSCACertificateFile: path
```

path 为包含 CA 证书的文件，或者如果使用 Mozilla NSS 的话，可以是证书名称。

olcTLSCACertificatePath

olcTLSCACertificatePath 命令指定在不同文件中包含单独 CA 证书存放的目录。该目录必须由 OpenSSL c_rehash 进行管理，生成指向实际证书文件的链接，一般而言，常使用 olcTLSCACertificateFile 作为替代。

假如 Mozilla NSS 被使用，olcTLSCACertificatePath 接受 Mozilla NSS 数据库路径（就像下例所说）。在这种情况下，c_rehash 是不需要的。

使用如下

```
olcTLSCACertificatePath: path
```

例如：使用 olcTLSCACertificatePat Mozilla NSS

通过 Mozilla NSS，olcTLSCACertificatePath 指定目录路径，该目录包含 NSS 证书和数据库文件

```
olcTLSCACertificatePath: sql:/home/nssdb/sharednssdb
```

certutil 命令用来给 NSS 数据库文件添加 CA 证书

```
certutil -d sql:/home/nssdb/sharednssdb -A -n "CA_certificate" -t CT,, -a -i certificate.pem
```

以上的命令添加了一个 CA 证书，以 PEM-formatted 格式保存，名字为 certificate.pem。-d 选项指定数据库目录，该目录包含 NSS 证书和数据库文件，-n 选项设置证书名称，-t CT,，意思是证书是可信任的，在 TLS 客户端和服务器端使用。-A 添加已存在的证书至证书数据库中，-a 允许使用 ASCII 格式作为输入输出，-i 将 certificate.pem 输入传递给命令。

olcTLSCertificateFile

olcTLSCertificateFile 命令指定包含 slapd 服务器证书的文件。

```
olcTLSCertificateFile: path
```

path 为包含 slapd 服务器证书的文件，如果使用 Mozilla NSS，改为证书名称。

例如：通过 Mozilla NSS 使用 olcTLSCertificateFile

当使用 Mozilla NSS 和证书关键数据库文件和 olcTLSCACertificatePath 命令，olcTLSCACertificatePath 用来指定证书的名称。首先，列出 NSS 数据库文件中可用的证书。

```
certutil -d sql:/home/nssdb/sharednssdb -L
```

选择一个证书，将它的名称传递给 olcTLSCertificateFile

```
olcTLSCertificateFile slapd_cert
```

olcTLSCertificateKeyFile

olcTLSCertificateKeyFile 命令指定文件，该文件包含私钥，私钥和存放在 olcTLSCertificateFile 的证书是匹配的。当前不支持加密的私钥，因此该文件必须被有效的保护。

```
olcTLSCertificateKeyFile: path
```

当使用 PEM 证书是时，path 替换为私钥文件路径。当使用 Mozilla NSS 时，path 替换为一个文件的名称，该文件包含 olcTLSCertificateFile 命令指定的证书的密码（参考下例使用 olcTLSCertificateKeyFile 和 Mozilla NSS）

例如： 使用 olcTLSCertificateKeyFile 和 Mozilla NSS

当使用 Mozilla NSS 时，该命令指定一个文件的名称，该文件包含 olcTLSCertificateFile 指定的证书的 key 的密码。

```
olcTLSCertificateKeyFile: slapd_cert_key
```

modutil 命令能够用来转变密码保护或改变 NSS 数据库文件密码

```
modutil -dbdir sql:/home/nssdb/sharednssdb -changepw
```

客户端配置

配置文件/etc/openldap/ldap.conf 在系统中是全局的，也有单独的用户在~/.ldaprc 配置中进行覆盖配置。

相同的指令可以创建一个 SSL 连接。在 OpenLDAP 命令比如 ldapsearch ，ldaps://字符串必须替代 ldap://。这些命令使用服务器端默认的 SSL 端口 636。

TLS_CACERT

TLS_CACERT 命令指定一个文件，包含客户端识别的所有的证书。该功能等同于服务器的 olcTLSCACertificateFile 命令。TLS_CACERT 应该在文件/etc/openldap/ldap.conf 中 TLS_CACERTDIR 选项前被指定。

```
TLS_CACERT path
```

path：CA 证书文件路径

TLS_CACERTDIR

TLS_CACERTDIR 命令指定在不同文件中包含单独 CA 证书存放的目录。跟 olcTLSCACertificatePath 命令类似。该目录必须由 OpenSSL c_rehash 进行管理，接受 Mozilla NSS 数据库文件路径，在这种情况下，c_rehash 是不需要的。

```
TLS_CACERTDIR directory
```

directory：包含 CA 证书的目录路径。使用 Mozilla NSS 时，为证书或关键数据库文件路径。

TLS_CERT

TLS_CERT 指定文件，包含客户端证书。该命令只有在用户~/.ldaprc 文件中被指定。在 Mozilla NSS 下，该命令指定的是证书的名字，由 TLS_CACERTDIR 命令指定。

```
TLS_CERT path
```

path：客户端证书文件路径，或者是 NSS 数据库证书的名称

TLS_KEY

TLS_KEY 指定包含私钥的文件，私钥是匹配存放在 TLS_CERT 指定的证书。该功能和服务器 olcTLSCertificateFile 类似，加密的文件是不支持的，所以该文件需要被小心保护，该选项只能在用户的~/.ldaprc 文件指定。

当使用 Mozilla NSS 时，TLS_KEY 指定一个文件，包含私钥的密码，用来保护 TLS_CERT 指定的证书。功能和 olcTLSCertificateKeyFile 类似，您可以使用 modutil 命令管理密码。

TLS_KEY 使用如下

```
TLS_KEY path
```

path：客户端证书文件路径，或者是 NSS 数据库中的密码文件名称。

4.2.3.5. 设置备份

备份是一个拷贝更新的进程，从一个 LDAP 服务器（provider）至一个或多个其它的服务器或客户端（consumer）。一个 provider 备份目录更新至 consumers，接收到的更新能够被 consumer 传播至其它的服务器端，因此一个 consumers

可以被当做一个 provider。一个 consumers 可以不是一个 LDAP 服务器端，它可以仅仅是一个个客户端。在 OpenLDAP，有多种备份模式，常见的有 mirror 和 sync。

为了使用选择好的备份模式，需要在 provider 和 consumers 的/etc/openldap/slapd.d/选择以下其中一种模式：

olcMirrorMode

olcMirrorMode 使用 mirror 备份模式

```
olcMirrorMode  on
```

其中 serverID 必须与 syncrepl 一起被指定。

olcSyncrepl

olcSyncrepl 使用 sync 备份模式

```
olcSyncrepl  on
```

4.2.3.6. 加载模块和后端

您可以使用动态加载模块用来增强 slapd 服务。在配置 slapd 的时候，可以使用--enable-modules 选项来配置那些可以支持的模块。模块保存在.la 结尾的文件中。

```
module_name.la
```

后端存储和数据检索应对 LDAP 请求。后端静态的编译至 slapd 或者当模块是被支持的，它们能够被动态的加载。对于后者，以下为命名约定

```
back_backend_name.la
```

为了加载模块和后端，在/etc/openldap/slapd.d/选择：

olcModuleLoad

olcModuleLoad 指定动态加载的模块

```
olcModuleLoad:  module
```

module：一个文件包含模块或后端，将要被加载。

4.2.4. 使用 LDAP 应用的 SELinux 策略

SELinux 是一个在 linux 内核中实现的一个强制访问控制机制。默认的，SELinux 会组织应用访问 OpenLDAP 服务器。为了允许应用通过 LDAP 验证，SELinux 的 allow_ypbind 必须被设为可用的。某些应用也需要 authlogin_nsswitch_use_ldap 是被允许的。以下是激活命令：

```
~]# setsebool -P allow_ypbind=1
~]# setsebool -P authlogin_nsswitch_use_ldap=1
```

-P 选项是是这次设置在系统重启一直保持有效。

4.2.5. 运行 OpenLDAP 服务

该章节描述了怎样启动、停止、重启和检查当前 LDAP 服务的状态。

4.2.5.1. 启动服务

使用 root 权限，开启 slapd 服务

```
~]# systemctl start slapd.service
```

使用 root 权限，设置开机启动 slapd 服务

```
[root@ft4 slapd.d]# systemctl enable slapd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/slapd.service
to /usr/lib/systemd/system/slapd.service.
```

4.2.5.2. 停止服务

停止服务

```
~]# systemctl stop slapd.service
```

设置开机不启动服务

```
[root@ft4 slapd.d]# systemctl disable slapd.service
Removed symlink /etc/systemd/system/multi-user.target.wants/slapd.service
```

4.2.5.3. 重启服务

重启服务

```
~]# systemctl restart slapd.service
```

该命令会先停止服务，马上再重启服务。使用该命令重新加载配置。

4.2.5.4. 检查运行状态

检查 slapd 服务运行状态

```
4.2.5.5. ~]$ systemctl is-active slapd.service
4.2.5.6. active
```

4.2.6. 配置系统使用 OpenLDAP 作为验证

为了配置系统使用 OpenLDAP 作为验证，确保所有的安装包在 LDAP 服务器和客户端机器上已经安装。怎么安装请参考章节 4.2.2 安装 OpenLDAP 组件和 4.2.3 配置 OpenLDAP 服务器。在客户端上，输入命令如下：

```
~]# yum install openldap openldap-clients nss-pam-ldapd
```

4.2.6.1. 迁移旧的验证信息至 LDAP 格式

迁移工具包提供了许多 shell 和 perl 脚本，可以帮助迁移旧的验证信息至 LDAP 格式。安装这些包，命令如下：

```
~]# yum install migrationtools
```

安装完后，脚本存放目录：/usr/share/migrationtools/。编辑/usr/share/migrationtools/migrate_common.ph 文件，修改以下行内容用来映射当前域。

```
# Default DNS domain
$DEFAULT_MAIL_DOMAIN = "example.com";
# Default base
$DEFAULT_BASE = "dc=example,dc=com";
```

或者，使用命令行指定环境变量。例如，运行 migrate_all_online.sh 脚本，使用默认的基准，设置 dc=example,dc=com

```
~]# export DEFAULT_BASE="dc=example,dc=com" \
/usr/share/migrationtools/migrate_all_online.sh
```

决定使用哪个脚本来运行迁移用户数据库，参考表格 4-10 常用的 LDAP 迁移脚本。

表格 4-10 常用的 LDAP 迁移脚本

已存在的服务	LDAP 是否运行	使用的脚本
/etc flat files	yes	migrate_all_online.sh
/etc flat files	no	migrate_all_offline.sh
NetInfo	yes	migrate_all_netinfo_online.sh
NetInfo	no	migrate_all_netinfo_offline.sh
NIS (YP)	yes	migrate_all_nis_online.sh
NIS (YP)	no	migrate_all_nis_offline.sh

怎么使用这些脚本，参考 README 和 migration-tools.txt 文件，存放在/usr/share/doc/migrationtools-47/目录下。

4.3. 文件和打印服务器

这个章节主要介绍 Samba 的安装和配置，一个 Server Message Block (SMB)和 common Internet file system (CIFS)协议的开源实现，vsftpd，中标麒麟高级服务器操作系统软件的 FTP 服务器。此外，还介绍了怎么使用打印服务器工具去配置打印机。

4.3.1. Samba

Samba 是标准 linux 开源 windows 套件项目。它实现了 SMB 和 CIFS 协议。它允许不同的系统包括 Microsoft Windows®, Linux, UNIX 等，访问基于 windows 的文件和打印机共享。Samba 的 SMB 使用类似 windows 服务器与 windows 客户端一样。

samba 安装

```
~]# yum install samba
```

4.3.1.1. Samba 介绍

Samba 是一个很重要的组件，作为无缝集成 Linux 服务器和桌面至 Active Directory (AD)环境。它可以作为一个域控制器，或者是常规的域成员。

Samba 能够做什么：

- 服务目录树和 Linux，Unix 和 windows 客户端的打印机
- 协助网络浏览
- Windows 域进行身份验证登录
- 提供 Windows Internet Name Service (WINS)名称服务解决方案
- Windows NT®-style Primary Domain Controller (PDC)
- Backup Domain Controller (BDC) for a Samba-based PDC
- 域名服务器成员
- 加入 Windows NT/2000/2003/2008 PDC/Windows Server 2012

Samba 不能做什么：

- 作为 BDC 替代 windows PDC
- 作为域名服务器控制器

4.3.1.2. Samba 以及相关服务

Samba 是由三个守护进程组成（smbd，nmbd 和 winbindd）。三个服务（smb，nmb 和 winbind）控制着守护进程的启动，停止和其它相关服务功能。这些服务作为不同的 init 脚本。以下详细介绍每一个守护进程。

smbd

服务器端 smbd 守护进程给 windows 客户端提供文件共享和打印服务。另外，它负责用户身份验证，资源锁定，数据共享通过 SMB 协议。默认的，服务器监控 SMB 传输端口为 TCP 端口 139 和 445。smbd 守护进程是被 smb 服务控制。

nmbd

nmbd 守护进程对于 NetBIOS 名称服务的请求（SMB/CIFS in Windows-based systems）进行理解和响应。这些系统包括 Windows 95/98/ME, Windows NT, Windows 2000, Windows XP, 和 LanManager 客户端。默认服务器监控 NMB 传输端口是 UDP 端口 137。

winbindd

winbind 服务解决了用户和组信息从运行在 Windows NT, 2000, 2003, Windows Server 2008, or Windows Server 2012 服务器上的接收问题。这使得 windows 用户和组信息能够被 UNIX 平台识别。这是被 Microsoft RPC calls，Pluggable Authentication Modules (PAM)和 the Name Service Switch (NSS)所实现。这允许 Windows NT 域和 Active Directory 用户被当做 UNIX 用户进行操作。虽然和 Samba 进行了捆绑，但是 winbind 服务是和 smb 分开进行控制的。

连接 Samba 共享

您可以使用 **Nautilus** 或命令行连接可用的 Samba 共享。

挂载共享

有时候，需要对 Samba 共享进行挂载操作，需要使用 root 用户登录，命令如下：

```
mount -t cifs //servername/sharename /mnt/point/ -o username=username,password=password
```

该命令将 sharename 挂载至本地 /mnt/point/目录

4.3.1.3. 配置 Samba 服务器

默认的配置文件/etc/samba/smb.conf 允许用户访问它们的 home 目录作为 samba 共享。它可以共享所有配置好的打印机作为 samba 共享打印机。您可以在系统中附加一个打印机并且通过 windows 机器打印东西。

图形配置

配置 Samba 使用图形接口，可以参考 http://www.samba.org/samba/GUI/。

命令行配置

Samba 使用/etc/samba/smb.conf 作为配置文件。修改这个配置文件后，需要重启 Samba 才能够生效

```
~]# systemctl restart smb.service
```

指定 windows 工作组和 Samba 服务器简短的描述，编辑/etc/samba/smb.conf 文件

```
workgroup = WORKGROUPNAME
server string = BRIEF COMMENT ABOUT SERVER
```

在 linux 系统上创建 Samba 共享目录，在/etc/samba/smb.conf 文件中添加以下内容：

例如：Samba 服务器的配置

```
[sharename]
comment = Insert a comment here
```

```
path = /home/share/
valid users = tfox carole
writable = yes
create mask = 0765
```

该例子允许用户 tfox 和 carole 通过 Samba 客户端对 Samba 服务器/home/share/目录进行读写操作

加密密码

加密密码是可以被使用的，因为加密的密码会更安全。创建一个使用加密密码的用户，命令如下

```
smbpasswd -a username
```

4.3.1.4. 启动和关闭 Samba

启动

```
~]# systemctl start smb.service
```

关闭

```
~]# systemctl stop smb.service
```

重启

```
~]# systemctl restart smb.service
```

condrestart（特定条件下重启）在当前正在运行的情况下才会启动 smb。这个选项对脚本是非常有用的，当守护进程没有运行的时候，将不会被启动。

```
~]# systemctl try-restart smb.service
```

重新载入/etc/samba/smb.conf 配置文件，不需要进行服务的重启，命令如下

```
~]# systemctl reload smb.service
```

开机自启动

```
~]# systemctl enable smb.service
```

4.3.1.5. Samba 安全模式

Samba 有两种安全模式，share-level 和 user-level。share-level 已经被弃用了，User-level 可以有三种不同的实现方式，这些方式被称作安全模式。

User-Level 安全

User-level security 是 Samba 默认推荐的配置。即使 security = user 没有在/etc/samba/smb.conf 配置文件中列出，它照样会被使用。当服务器接收了客户端的用户名和密码，客户端在挂载共享的时候就能够不需要指定密码。Samba 也能够接收到基于用户名和密码的请求。客户端通过使用一个单独的 UID 维持已验证的状态。

/etc/samba/smb.conf 文件中，security = user 设置 user-level security

```
[GLOBAL]
...
security = user
...
```

Samba Guest 共享

上面已经提到过，share-level security 模式已经被放弃。怎么配置 Samba guest 共享，不使用 security = share 选项。参考以下步骤：

实例：配置 Samba Guest 共享

1) 创建用户映射文件，例如/etc/samba/smbusers，添加以下行：

```
nobody = guest
```

2) 修改/etc/samba/smb.conf，不要使用 valid users

```
[GLOBAL]
...
security = user
map to guest = Bad User
username map = /etc/samba/smbusers
```

```
...
```

username map：1 步骤指定

3) 修改/ect/samba/smb.conf，不要使用 valid users

```
[SHARE]
...
guest ok = yes
...
```

下面将会介绍其它 user-level security 实现方式。

Domain Security Mode (User-Level Security)

在这种方式下，Samba 服务器有一个机器账号（domain security 信任的账号），所有验证的请求都需要通过域控制器。Samba 服务器要作为域成员，需要配置/etc/samba/smb.conf。

```
[GLOBAL]
...
security = domain
workgroup = MARKETING
...
```

Active Directory Security Mode (User-Level Security)

假如您有一个 Active Directory 环境，加入域作为一个本地成员是可能的。即使安全策略限制使用 NT-compatible 验证协议，Samba 服务器可以通过使用 Kerberos 加入 ADS。Samba 在 Active Directory member 模式下可以接受 Kerberos tickets。

修改/etc/samba/smb.conf

```
[GLOBAL]
...
security = ADS
realm = EXAMPLE.COM
```

```
password server = kerberos.example.com
...
```

Share-Level 安全

在该模式下，服务器从客户端那仅接收一个没有明确用户名的密码。服务器期望一个密码可以给所有的共享，不依赖用户名。许多报告指出，Microsoft Windows 客户端兼容 share-level security 服务器。该模式已经被 Samba 抛弃。配置项 security = share 必须被升级成 user-level security。如果想使用，请参考 User-Level 安全中的例子：配置 Samba Guest 共享。

4.3.1.6. Samba 网络浏览

网络浏览使得 Windows 和 Samba 服务器出现在 Windows Network Neighborhood 中，在 Network Neighborhood 里面，服务器有自己的图标，打开图标，服务器可用的共享和打印机可以被看到。

网络浏览需要 NetBIOS 通过 TCP/IP。NetBIOS-based 网络使用 UDP 发送消息完成浏览列表管理。没有 NetBIOS 和 WINS 作为 TCP/IP 主机名称解决方案基本的方法，其它的例如静态文件（/etc/hosts）或 DNS 是必须要使用的。

一个域的主浏览服务器从所有子网中的本地主浏览服务器收集核对所有的浏览列表，因此在组和子网间可以相互浏览。

域浏览

默认的，一个 Windows 服务器 PDC 作为一个域，也是该域的主浏览服务器。Samba 服务器不应该被作为一个主浏览服务器。

在许多子网中，不包含 Windows 服务器 PDC，因此，Samba 服务器可以作为一个本地浏览服务器。配置/etc/samba/smb.conf 文件作为一个本地主浏览服务器（或不作为），配置过程和组配置类似（参见 4.3.1.3 配置 Samba 服务器。）

WINS（Windows Internet Name Server）

Samba 和 Windows NT 服务器可以被作为一个 WINS 服务器。当 WINS 服务器和 NetBIOS 一起使用时，UDP 传播能够被转发在网络中允许使用名称转换。没有 WINS 服务器时，UDP 传播只能在当前子网传播不能够被转发至其它子网，组和域。假如需要 WINS 备份功能，不要使用 Samba 作为您的 WINS 服务器，因为 Samba 不支持 WINS 备份。

在一个 NT/2000/2003/2008 服务器和 Samba 混合环境中，推荐使用 Microsoft WINS。在一个只有 Samba 环境中，推荐使用 Samba 作为 WINS。

下面是一个使用 Samba 作为 WINS 服务器的例子。

例如：配置 WINS 服务器

```
[global]
wins support = yes
```

4.3.1.7. Samba Distribution Programs

net

```
net <protocol> <function> <misc_options> <target_options>
```

net 命令的使用和在 Windows 和 MS-DOS 系统中类似。第一个参数指定命令使用的协议。protocol 选项可以是 ads，rap 或 rpc。Active Directory 使用 ads，Win9x/NT3 使用 rap，Windows NT4/2000/2003/2008 使用 rpc。假如 protocol 选项没指定，net 会自动尝试确定它。

下面例子显示了主机名为 wakko 的可用的共享：

```
~]$ net -l share -S wakko
Password:
```

下面例子显示了 wakko 主机可用的 Samba 用户列表

```
~]$ net -l user -S wakko
root password:
```

nmblookup

```
nmblookup <options> <netbios_name>
```

nmblookup 可以解决 NetBIOS 名称转换为 IP 地址。该项目在子网中会广播查询直到有目标主机回应。

下面的例子是显示 NetBIOS 名称 trek 的 IP 地址：

```
~]$ nmblookup trek
querying trek on 10.1.59.255
10.1.56.45 trek<00>
```

pdbedit

```
pdbedit <options>
```

pdbedit 项目管理 SAM 数据库存储的账户。所有的后端包括支持 smbpasswd，LDAP 和 tdb 数据库

下面的例子是添加，删除和列出用户

```
~]$ pdbedit -a kristin
new password:
retype new password:
~]$ pdbedit -x joe
~]$ pdbedit -L
andriusb:505: lisa:504: kristin:506:
```

rpcclient

```
rpcclient <server> <options>
```

rpcclient 项目问题管理命令使用 Microsoft RPCs，这个是给知道 Microsoft RPCs 复杂性用户使用的。

smbcacls

```
smbcacls <//server/share> <filename> <options>
```

smbcacls 项目修改 Windows ACLs 文件和 Samba 共享的目录或者是 Windows 服务器。

smbclient

```
smbclient <//server/share> <password> <options>
```

smbclient 是 UNIX 系统通用的客户端，功能和 ftp 类似。

smbcontrol

```
smbcontrol -i <options>
```

```
smbcontrol <options> <destination> <messagetype> <parameters>
```

smbcontrol 项目发送控制消息来运行 smbd，nmbd 或 winbindd 守护进程。smbcontrol –i 交互式运行，直到出现空行或者‘q’被输入

smbpasswd

```
smbpasswd <options> <username> <password>
```

smbpasswd 项目管理加密密码。该项目能够被超级用户修改所有的用户密码还有普通用户修改自己的 Samba 密码。

smbspool

```
smbspool <job> <user> <title> <copies> <options> <filename>
```

smbspool 项目是 Samba 一个 CUPS 兼容的打印接口。虽然被设置为 CUPS 打印机，smbspool 可以和非 CUPS 打印机一起使用。

smbstatus

```
smbstatus <options>
```

smbstatus 项目显示当前 Samba 连接状态

smbtar

```
smbtar <options>
```

smbtar 程序执行基于 Windows 共享文件和目录的备份和恢复。虽然和 tar 命令相似，两者不兼容。

testparm

```
testparm <options> <filename> <hostname IP_address>
```

testparm 程序检查/etc/samba/smb.conf 文件的语法，假如您的 smb.conf 存储在默认位置（/etc/samba/smb.conf），则不需要指定。指定主机名和 IP 地址给 testparm 程序检查 hosts.allow 和 host.deny 文件是否配置正确。testparm 程序可以显示 smb.conf 文件和服务器规则在测试后。在调试的时候可以给丰富经验的管理员提供有用的信息。例如

```
~]$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[tmp]"
Processing section "[html]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
<enter>
# Global parameters
[global]
workgroup = MYGROUP
server string = Samba Server
security = SHARE
log file = /var/log/samba/%m.log
max log size = 50
socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
dns proxy = no
[homes]
comment = Home Directories
read only = no
browseable = no
[printers]
comment = All Printers
path = /var/spool/samba
printable = yes
browseable = no
[tmp]
comment = Wakko tmp
path = /tmp
guest only = yes
```

```
[html]
comment = Wakko www
path = /var/www/html
force user = andriusb
force group = users
read only = no
guest only = yes
```

wbinfo

```
wbinfo <options>
```

wbinfo 程序显示 winbindd 守护进程信息。winbindd 进程必须是运行的。

4.3.1.8. 其他资源

安装文档

/usr/share/doc/samba-<version-number>/

可以查看 man 手册

- smb.conf(5)
- samba(7)
- smbd(8)
- nmbd(8)
- winbindd(8)

有用的网站

- http://www.samba.org/
- https://wiki.samba.org/index.php/User_Documentation
- http://samba.org/samba/archives.html

4.3.2. FTP

该章节将会介绍 FTP 协议以及 vsftpd（Linux 系统 FTP 服务器）

FTP 使用客户端-服务端架构模式来传输文件，使用 TCP 网络协议。因为 FTP 是一个较早的协议，他不支持加密用户和密码进行验证。基于这个原因，FTP 是被认为不安全的传输协议，除非特殊情况下不建议使用。因为 FTP 在网络上

是非常流行的，它经常被用来分享文件。因此，作为一个管理员，需要知道它的独特性。

这个章节描述了怎么配置 vsftpd 来建立安全连接通过 TLS 和怎么通过 SELinux 来加强 FTP 的安全性。FTP 的一个更好的取代品是 sftp，是由 OpenSSH 组件工具提供。要想获取更多 OpenSSH 配置以及 SSH 协议，请参考 3.2OpenSSH。

不同于其它协议，FTP 需要多个端口才能正常工作。当 FTP 客户端建立一个到 FTP 服务器端的连接，它会打开端口 21 在服务器端——控制端口。这个端口被用来发送命令至服务器。所有从服务器端到客户端的数据请求是由专门的数据端口传输。数据连接端口，以及数据连接的初始化依赖于客户端请求数据方式：active 或 passive 模式。

active mode

Active mode 是 FTP 协议传输数据协议使用到的最原始的模式。当一个 active-mode 数据传输被 FTP 客户端初始化，服务器端会打开一个 20 端口给 IP 地址，和一个随机无特权的端口（大于 1024）。这样的安排意味着客户端机器必须被允许接受任何超过 1024 的端口连接。随着不安全网络的增长，使用防火墙来保护客户端机器现在是普遍的。因为这些客户端防火墙常常否定从主动模式传入的连接，所以 passive mode 被设计出来。

passive mode

像 active mode 一样，Passive mode 是被 FTP 客户端初始化的。当从服务器请求数据,FTP 客户端表明它想在 Passive mode 下访问数据，服务器在服务器上会提供一个 IP 地址和一个随机、无特权的端口(大于 1024)。然后客户端连接到该端口在服务器上下载请求的信息。

虽然 Passive mode 确实解决为客户端防火墙干扰数据连接问题，它可以视为管理服务器端防火墙。您可以在一个服务器上通过限制的范围无特权的端口在 FTP 服务器上减少开放端口的数量。这也简化了服务器配置防火墙规则的过程。

4.3.2.1. vsftpd 服务器

vsftpd 设计为快、稳定和安全。vsftpd 是中标麒麟高级服务器操作系统软件中的 FTP 服务器。是因为其能够处理大量的连接数和安全性。

vsftpd 所使用的安全模型有三个主要方面：

- 强大的特权和非特权分离过程——独立的进程处理不同的任务,每一个进程运行的任务只需要最小权限
- 需要提升权限的任务可以使用最小权限进行处理——利用 libcap 库兼容性的特性，需要 root 特权权限的任务可以用很小特权的进程执行
- 大多数进程运行在 chroot 下——只要可能，进程可以改变程序执行时参

考的根目录位置为当前共享的目录。这个目录被认为是 chroot 目录。例如，假如/var/ftp/目录是共享目录，vsftpd 指定/var/ftp/为新的 root 目录，作为/。这会减少黑客的攻击。

使用这些安全措施会对 vsftpd 如何处理请求产生影响，如下：

- 父进程运行时仅需最少的特权——父进程能够动态的计算最小特权等级将风险最小化。子进程处理直接与客户端交互和尽可能使用无权限运行。所有的需要特权的操作被一个小的父进程处理——就像 Apache HTTP Server 一样，vsftpd 分发无特权的子进程来处理传入的连接。这允许特权,父进程尽可能小处理任务。
- 所有从非特权子进程来的请求将会被父进程怀疑——和子进程进行对话是通过 socket，任何来自子进程的信息将会被检查。
- 大部分和 FTP 客户端交互式操作是在 chroot 环境下——因为子进程是非特权的，它们只有在共享目录下有登入权限，只允许攻击者能够访问共享目录。

启动和停止 vsftpd

以 root 用户登录

启动

```
~]# systemctl start vsftpd.service
```

停止

```
~]# systemctl stop vsftpd.service
```

重启

```
~]# systemctl restart vsftpd.service
```

有条件重启，当它已经在运行的时候，才能够执行

```
~]# systemctl try-restart vsftpd.service
```

设置开机自启动

```
~]# systemctl enable vsftpd.service
```

```
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
```

开启 vsftpd 多路拷贝

有时候，一台计算机会被用来作为多路 FTP 域。这种技术叫做多归属。vsftpd 的多归属的用法是运行多个进程的拷贝，每一个进程拥有自己独立的配置文件。

首先，给所有的网络设备分配好 IP。

再次，FTP 的域的 DNS 服务器必须对应正确的机器。

当 vsftpd 响应不同 IP 的请求时，进程的多路拷贝必须是运行的。为了分发 vsftpd 进程的多路实例，一个特殊的服务 systemd service unit (vsftpd@.service)是被 vsftpd 包提供的。

为了使用该服务，一个单独的 vsftpd 配置文件为各个 FTP 服务器实例是需要的，保存在/etc/vsftpd/目录。这些配置文件必须拥有独立的名称（例如/etc/vsftpd/vsftpd-site-2.conf），并且拥有 root 的读写权限。

每一个配置文件，要有如下参数作为监听 IPv 网络

```
listen_address=N.N.N.N
```

N.N.N.N 为 FTP 站点的 IP 地址。假如使用的是 IPv6，则使用 listen_address6

假如配置文件已经存放在/etc/vsftpd/目录，单独的 vsftpd 进程，可以由以下命令启动

```
~]# systemctl start vsftpd@configuration-file-name.service
```

在以上的命令中，修改 configuration-file-name 为需要的配置文件名称。例如 vsftpd-site-2，注意，.conf 后缀是不需要添加进来的。

如果想一次性启动多个 vsftpd 进程

```
~]# systemctl start vsftpd.target
~]# systemctl enable vsftpd.target
ln -s '/usr/lib/systemd/system/vsftpd.target' '/etc/systemd/system/multi-user.target.wants/vsftpd.target'
```

使用 TLS 加密 vsftpd 连接

为了对抗 FTP 固有的不安全性（使用明文传输）。vsftpd 进程可以使用 TLS 来对连接和传输进行加密。FTP 客户端必须支持 TLS。

在配置文件中 vsftpd.conf 设置 ssl_enable 为 YES 打开 TLS 的支持。

例如：配置 vsftpd 使用 TLS

vsftpd.conf 配置文件

```
ssl_enable=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
```

重启 vsftpd service 生效

```
~]# systemctl restart vsftpd.service
```

可以查看 vsftpd.conf(5)的 man 手册了解更多相关信息

vsftpd 的 SELinux 策略

SELinux 策略管理 vsftpd 守护进程(以及其他 ftpd 流程)，定义了一个强制访问控制，为了允许 FTP 守护进程访问特定文件或目录，需要分配给他们适当的标签。

例如，为了能够匿名共享文件，public_content_t 标签必须分配给共享的文件和目录。您可以使用 chcon 命令

```
~]# chcon -R -t public_content_t /path/to/directory
```

/path/to/directory 是您想分配标签的目录路径，如果您想建立一个上传文件的目录，您必须分配一个特使的标签 public_content_rw_t。除此之外，allow_ftpd_anon_write 选项必须设置为 1，使用 setsebool 命令设置

```
~]# setsebool -P allow_ftpd_anon_write=1
```

假如您想让本地用户通过 FTP 访问 home 目录，在中标麒麟高级服务器操作系统软件中这个是缺省的设置，ftp_home_dir 选项必须设置为 1。假如 vsftpd 允许以独立模式运行，在中标麒麟高级服务器操作系统软件中这个是缺省的设置，ftpd_is_daemon 必须设置为 1。

4.3.3. 打印设置

打印设置工具用来打印机的配置，维护打印机配置文件，打印排队目录以及打印过滤和打印机管理类。

该工具是基于通用 Unix 印刷系统(CUPS)。如果你升级的系统是先前的中标麒麟高级服务器操作系统软件版本，使用的是 CUPS，在升级过程将会保存打印机信息。

启动打印机配置工具

从命令行启动打印机配置工具，输入 system-config-printer，打印机配置工具启动。或者是，当你使用 GNOME 桌面，点击**【应用程序】→【系统工具】→【设置】→【设备】→【打印机】**，弹出用户配置对话框；输入登录用户面解锁打印机配置界面。

打印机配置工具启动如下图。

图 4-10 打印设置

启动打印机设置

打印机安装进程依赖于打印机队列类型。

假如你是设置一个通过 USB 连接的本地打印机，打印机会自动发现和添加。系统将提示您确认要安装的软件包,并提供一个管理员或 root 用户密码。本地打印机连接与其他端口类型和网络打印机需要手动设置。

以下是设置一个手动打印机的步骤：

1) 启动打印机配置工具（参见启动打印机配置工具）。

2) 在验证对话框中，输入登录用户密码。假如这是你第一次配置一个远程的打印机，你将会被提示授权防火墙操作。

3) 选择打印机连接类型,在右边的区域提供细节。

添加一个本地打印机

添加本地打印机通过串行端口连接，步骤如下：

1) 打开添加打印机对话框（参考启动打印机设置）。

2) 假如设备没有自动出现，在左边的列表，选择打印机连接的端口（例如 **Serial Port #1** 或者 **LPT #1**）

3) 在右边,进入连接属性。

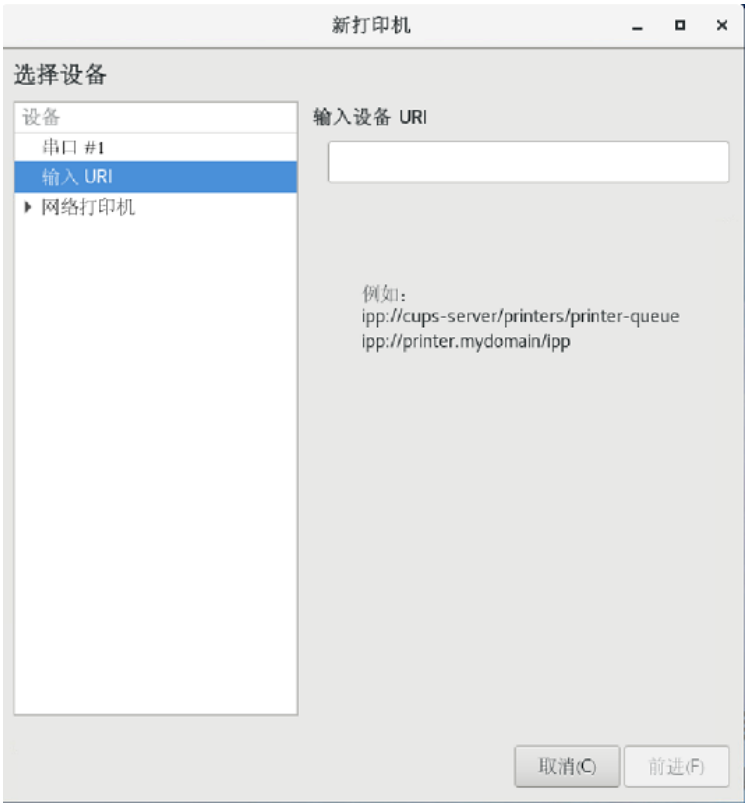


图 4-11 添加一个本地打印机

4) 点击【前进】，按步骤完成打印机设置即可。

添加 URI 打印机

步骤如下：

1）打开添加打印机对话框（参见启动打印机配置工具）。

2）在左边的列表，选择“输入 URI”输入需添加打印机正确的 URI 地址。

3）点击“前进”，按步骤配置即可完成。

添加网络打印机

添加步骤如下：

1）打开添加打印机对话框（参见启动打印机设置）。

2）在左边的设备列表里面，点击“网络打印机”，展开后可看到选项：

查找网络打印机、Internet Printing Protocol（IPP）、APPSocket/HP JetDirect、互联网打印协议（IPP）、互联网打印协议（https）、LPD/LPR 主机或者打印机、使用 SAMBA 的 Windows 打印机。配置网络打印机需要在防火墙配置中开启相应的服务及端口。

3）按您的实际需要选择添加的网络打印机类型。

4）在右侧，输入相应的正确设置信息，点击“前进”。

5）按提示完成相应的配置即可完成打印机的添加。

图 4-12 添加网络打印机

打印测试页

设置打印机或修改打印机配置后,打印测试页,确保打印机正常工作：

1) 打印机在打印窗口中点击鼠标右键,选择“属性”。

2) 在属性窗口中,点击左边的设置。

3) 在显示设置选项卡上,单击打印测试页按钮。

修改现有的打印机

删除现有的打印机,在打印设置窗口,选择打印机和打印机→删除。确认打印机删除。另外，按删除键。

设置默认打印机,在打印机列表中右键单击打印机,然后单击上下文菜单的设置为默认按钮。

设置页面

改变打印机驱动程序配置,双击打印机列表中对应的名称并单击左边的设置标签显示的设置页面。

您可以修改打印机设置例如打印机型号，打印测试页,更改设备位置(URI),等等。

共享打印机

在策略页面，你可以标志一个打印机作为共享：假如一个打印机是共享的，网络用户可以使用它。为了能够使用共享功能，选择共享打印机。

打印机选项页面

打印机选项页面包含各种配置选项的印刷媒体和输出,和它的内容可能会有所不同从打印机到打印机。它包含一般印刷、纸张、质量和印刷大小设置。

任务选项页面

在工作中选择页面上,您可以详细看到打印机工作的选项。点击左边的工作选择标签来显示页面。编辑默认设置应用自定义工作选项,如复制份数、定位、页面每一面，缩放(增加或减少可打印区域的大小,可以用来满足一个超大的打印区域到一个较小的物理表的打印介质),详细文本选项,选择和自定义工作。

墨水或碳粉水平页面

墨水/碳粉水平页面包含墨粉状态细节，是否可用和打印机状态信息。点击左边的墨水/碳粉水平标签来显示页面。

管理打印作业

当你发送一个打印作业到打印机，如从 Emacs 打印文本文件或打印一个图

像，打印的工作是添加到打印排队队列。打印排队队列是一个打印作业列表,显示为发送到打印机的每个打印请求信息，如请求的状态，工作数量等等。

在打印过程中，打印机状态图标出现在通知区域的面板。检查打印作业的状态，单击打印机地位，显示一个窗口。

取消、持有、发布、转载或验证一个打印作业，选择任务菜单，在 GNOME Print Status 选择任务，点击相应的命令。

使用 shell 命令查看打印任务列表，使用 lpstat -o。

例如：lpstat -o 的输出。

```
$ lpstat -o
Charlie-60    twaugh          1024    Tue 11 Feb 2020 16:42:11 GMT
Aaron-61      twaugh          1024    Tue 11 Feb 2020 16:42:44 GMT
Ben-62        root            1024    Tue 11 Feb 2020 16:45:42 GMT
```

如果你想取消打印作业,使用请求命令 lpstat -o 找到任务号,然后使用命令 cancel job number。例如 cancel 60 将取消打印作业在上面例子中“lpstat - o 输出”。你不能使用 cancel 命令取消别人启动的打印作业。然而，你可以强制删除任务，通过 cancel -U root job_number 命令，为了防止这样的删除，修改打印操作策略验证进行强制 root 验证。

你可以通过 shell 命令打印文件。例如 lp sample.txt，打印文件 sample.txt。

4.4. 使用 chrony 套件配置 NTP

在 IT 行业，保持精确的时间是非常重要的，这有很多原因。例如，在网络上，包分发和日志是需要精确的时间戳的。在 linux 系统中，NTP 协议由守护进程运行在用户空间实现。

用户空间的守护进程更新运行在内核空间的系统时钟。系统时钟能够使用多种时钟资源保持时间准确。通常的使用 Time Stamp Counter(TSC)。TSC 是一个 CPU 寄存器用来计算周期数。它非常快,高分辨率,没有中断。

有两个守护进程的选择，ntpd 和 chrony，来自 ntpd 和 chrony 包。本节描述和 chrony 套件的使用的实用程序来更新系统时钟系统,不符合传统的永久网络化。

4.4.1. chrony 套件介绍

Chrony 是运行在用户空间的守护进程的命令行程序，系统如果经常开关机，会花费很多时间通过 ntpd 校对系统时间。

4.4.1.1. ntpd 和 chronyd 的差异

最主要的差异是用于控制计算机的时钟的算法。chronyd 能够比 ntpd 做的更好。

- 当外部基准时间只能够间歇性的访问时 chronyd 能够工作的很好。ntpd 需要可持续查询时间基准才能够工作的好。
- chronyd 能够在网络拥堵的时候工作的好
- chronyd 通常可以同步时钟更快和更好的时间精度
- chronyd 能够在时钟的速度突然变化时迅速适应,例如,由于晶体振荡器的温度的变化,而 ntpd 可能需要很长时间才能再次适应下来。
- 在默认配置中，在系统启动后时间已经同步，chronyd 从未设置时间，为了不打乱其他运行程序。ntpd 也可以配置为不同步时间，但它必须使用不同的方式调整时钟,这会有一些缺点。
- 在 linux 操作系统上，chronyd 能够在一个很大的范围调整时间速率。这允许它在一个损坏或不稳定的机器上进行操作。例如，在一些虚拟机上。

chronyd 能够做一些 ntpd 不能做的事情。

- chronyd 支持孤立网络时间校正的唯一方法是手动输入。例如,通过管理员看时钟。chronyd 能够在不同的更新中检查出错误信息，评估计算机过多或丢失的时间速率。
- ntpd 支持 NTP 版本 4（RFC 5905），包括广播，多播，manycast 客户端和服务器，和孤儿模式。它还支持额外的身份验证方案基于公钥加密(RFC 5906)。chronyd 使用 NTP 版本 3（RFC 1305），兼容版本 4.
- ntpd 包括许多参考时钟的驱动而 chronyd 依赖于其他项目,例如 gpsd,访问数据的参考时钟。

4.4.1.2. NTP 守护进程的选择

- Chrony 可以考虑在这些系统中使用，这些系统会经常暂停或间歇地断开连接和重新连接网络，例如移动和虚拟机系统。
- NTP 守护进程（ntpd）应该考虑用在经常保持不变的系统上。系统需要使用广播或多播 IP，或执行身份验证数据包的 Autokey 协议，应该考虑使用 ntpd。chrony 仅仅支持对称密钥身份验证，使用 MD5 消息身份验证代码(MAC)，SHA1 或者更强的哈希算法。而 ntpd 还支持 Autokey 认证协议，该协议可以利用 PKI 系统。Autokey 在 RFC5906 中有描述。

4.4.2. 理解 CHRONY 及其配置

4.4.2.1. 理解 chronyd

chronyd 是 chrony 的守护进程，运行在用户空间，调整运行在内核的系统时

钟。它通过咨询外部时间源，使用 NTP 协议来进行调整。当外部引用并不可用，chronyd 将使用最后被计算存储在文件的数据。它还可以被 chronyc 手动进行修改。

4.4.2.2. 理解 chronyc

chronyd 是 chrony 的守护进程，可以被命令行组件 chronyc 控制。这个工具提供了一个命令提示符输入一些命令可以对 chronyd 进行更改。默认的配置 chronyd 仅仅接收本地的 chronyc 命令，chronyc 可以配置为 chronyd 可以接收外部控制。chronyc 可以远程运行后第一个配置 chronyd 接受远程连接。IP 地址允许连接到 chronyd 应严格控制。

理解 chrony 配置命令

chronyd 默认的配置文件/etc/chrony.conf，-f 选项可以指定一个配置文件的路径。

Comments

Comments should be preceded by #, %, ; or !

allow

可选择的，指定一个主机，子网或者网络连接一台可以作为 NTP 服务器的机器。默认是不允许连接。

例如：

```
allow server1.example.com
```

通过主机名，指定一个主机，运行连接

```
allow 192.0.2.0/24
```

指定一个网络允许连接

```
allow 2001:db8::/32
```

指定一个 IPv6 地址

cmdallow

该命令和 allow 命令相似，除了它允许特定的子网或主机的控制接入（而不是 NTP 客户端接入）。（控制接入，意思是 chronyc 能够运行在其它主机上并且能够通过本地计算机连接到 chronyd）他的语法是一样的。cmddeny all 命令和 c

mdallow all 命令类似。

dumpdir

保存 chronyd 服务重启的测量历史目录路径。

dumponexit

假如该命令是运行的，它表明 chronyd 必须为所有的时间源保存测量的历史。

local

local 关键字是允许 chronyd 通过客户端推送输入进行时间同步，即使它没有同步源。该选项经常被用来在 master 主机上使用，在一个独立的网络中，许多计算机需要同步，master 主机需要通过手动输入保持准确时间。

例如：

```
local stratum 10
```

log

log 命令表明某些信息将要被记录日记。它接收以下选项：

measurements

该选项记录了测量以及相关信息，生成 measurements.log 文件

statistics

该选项记录了回归处理相关信息，生成 statistics.log 文件

tracking

该选项记录了系统的收益或损失的估计，生成 tracking.log 文件

rtc

这个选项记录了系统的实时时钟信息

refclocks

这个选项记录了原始和过滤参考时钟测量，生成 refclocks.log。

tempcomp

这个选项记录温度测量和系统补偿速率，生成 tempcomp.log 文件

log 日志记录文件存放在 logdir 指定的目录

```
log measurements statistics tracking
```

logdir

指定日志文件存放的目录

```
logdir /var/log/chrony
```

makestep

通常，chronyd 将根据需求通过减慢或加速时钟，使得系统逐步纠正所有时间偏差。在某些特定情况下，系统时钟可能会漂移过快，导致该调整过程消耗很长的时间来纠正系统时钟。该指令强制 chronyd 在调整期大于某个阀值时步进调整系统时钟，但只有在因为 chronyd 启动时间超过指定限制（可使用负值来禁用限制），没有更多时钟更新时才生效。

```
makestep 1000 10
```

maxchange

该指令将指定最大修正偏移量，当偏移量大于指定的阀值，chronyd 将会放弃并且退出，将消息发送给 syslog

```
maxchange 1000 1 2
```

第一个时钟更新后，chronyd 将会检查每一个时钟偏移量的更新，将会忽略 2 次大于阀值的偏移量修正。

maxupdateskew

chronyd 的任务之一就是要相对于其源而言计算机的时钟运行的快或慢的程度。maxupdateskew 参数是确定估计是否是否可靠的阈值，缺省情况下，阈值是 1000ppm，语法格式如下：

```
maxupdateskew skew-in-ppm
```

noclientlog

这个命令没有参数，客户端不记录 log 日志

reselectdist

当 chronyd 选择同步源可用的来源,它将更喜欢最小的同步距离。然而,为了避免频繁的重新选择有来源相似的距离时,一个固定的距离被添加。默认情况下,距离是 100 微秒。

格式如下

```
reselectdist dist-in-seconds
```

stratumweight

stratumweight 指令设置当 chronyd 从可用源中选择同步源时，每个层应该添加多少距离到同步距离。默认情况下，CentOS 中设置为 0，让 chronyd 在选择源时忽略源的层级。

格式如下

```
stratumweight  dist-in-seconds
```

默认情况下，dist-in-seconds 为 1 秒

rtcfile

rtcfile 指令定义了一个文件的名称，该文件，chronyd 可以保存跟踪系统的实时时钟的准确性(RTC)参数。语法的格式是:

```
rtcfile  /var/lib/chrony/rtc
```

rtcsync

rtcsync 指令将启用一个内核模式，在该模式中，系统时间每 11 分钟会拷贝到实时时钟（RTC）。

chronyc 的安全性

和 Network Time Protocol (NTP) 相比，PTP 最主要的优点是硬件的支持，包括许多的 network interface controllers (NIC)和 network switches。极大的提高了时间同步的准确性。

4.4.3. 使用 chrony

4.4.3.1. 安装 chrony

使用 root 用户登录

```
~]# yum install chrony
```

默认的 chrony 进程位置/usr/sbin/chronyd。命令行组件安装在/usr/bin/chronyc。

4.4.3.2. 检查 chronyd 状态

```
~]$ systemctl status chronyd
```

4.4.3.3. 启动 chronyd

启动

```
~]# systemctl start chronyd
```

设置开机自启动

```
~]# systemctl enable chronyd
```

4.4.3.4. 关闭 chronyd

关闭

```
~]# systemctl stop chronyd
```

取消开机自启动

```
~]# systemctl disable chronyd
```

4.4.3.5. 检查 chrony 是否同步

为了检查 chrony 是否同步，使用 tracking，sources 和 sourcestats 命令。

检查 chrony Tracking

命令如下

```
~]$ chronyc tracking
```

字段如下：

Reference ID

服务器同步地址。假如是 127.127.1.1，表示该计算机是没有跟外部源进行同步，您正在使用 local 模式操作。

Stratum

拥有的层级数

Ref time

最后一次同步后测量的 UTC 时间

System time

系统时间

Last offset

最后一次预估的偏移量

RMS offset

平均偏移量

Frequency

如果 chronyd 没有进行纠错，系统时间出错的速率，例如：1ppm 意思是系统时间 1 秒，可能实际时间是 1.000001 秒。

Residual freq

显示当前选中源的剩余频率

Skew

frequency 的估计误差

Root delay

这是网络路径延迟到最终同步计算机时的 stratum-1 计算机的总和。在某些极端的情况下，此值可以为负。

检查 chrony 来源

命令如下

```
~]$ chronyc sources
```

M

表示源的模式。^表示一个服务器，=表示一个 peer，#表示本地连接的参考时钟

S

表示源的状态。*表示正在同步，+表示已连接，-表示已被排除，？表示连接已丢失。“x”表示一个时钟 chronyd 认为是 falseticker(与大多数其他来源的时间是不一致的),“~”表示一个源的时间似乎有太多的变化。

Name/IP address

源的 IP 地址或名称

Stratum

源的层，层 1 表明计算机附带本地参考时钟，与层 1 同步的计算机在层 2，与层 2 同步的计算机在层 3，以此类推。

Poll

显示源被 polled 的速率，例如值为 6 的时候，就表示每 64s 进行一次测量。

Reach

显示最后一个收到的源的时间。一般是在几秒钟之间。字母 m h d 或 y 显

示分钟,小时,几天或几年。10 年的值表示没有收到这个源。

LastRx

显示最后一个收到的源的时间。一般是在几秒钟之间。字母 m h d 或 y 显示分钟,小时,几天或几年。10 年的值表示没有收到这个源。

Last sample

此列显示本地时钟与最新的测量数据源之间的偏移量。

检查 chrony 来源统计

sourcestats 命令显示当前被检查的源的漂移速度和偏移量的信息。-v 参数能够被指定，意思是冗长的。在这种情况下，额外的行显示为一个提醒列。

```
~]$ chronyc sourcestats
```

Name/IP address

NTP 服务器的地址

NP

这是样本点的数量目前为服务器保留。漂移速率和电流偏移估计通过这些点进行线性回归。

NR

这是运行的 residuals 具有相同回归符号的数量。

Span

这是最古老的和最新的样本之间的时间间隔。如果没有显示单元的值，则表示是在几秒钟内。在这个例子中,间隔是 46 分钟。

Frequency

估计的剩余频率，在这种情况下，计算机的时钟估计是 $1/10^9$ 相对于服务器来说

Freq Skew

估计的误差界限

Offset

估计的偏移量

Std Dev

这是估计样本标准差

4.4.3.6. 手动调整系统时钟

命令如下

```
~]# chronyc
chrony> makestep
200 OK
```

4.4.4. 为不同的环境设置 chrony

4.4.4.1. 为一个很少连接的系统设置 chrony

这个例子是用于系统使用 dial-on-demand 连接。正常的配置应满足移动和虚拟设备连接断断续续。首先,审查和确认/etc/chrony.默认设置配置类似于以下几点:

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
```

command key ID 是在安装时候生成，应符合 commandkey 值，/etc/chrony.keys。

添加 NTP 服务器

```
server 0.pool.ntp.org offline
server 1.pool.ntp.org offline
server 2.pool.ntp.org offline
server 3.pool.ntp.org offline
```

4.4.4.2. 为一个独立网络系统设置 chrony

在一个从来不和外部网络进行连接的独立的网络中，可以选择一台计算机作为时间主机，其它主机作为主机的客户端。

在 master 主机上，编辑/etc/chrony.conf

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0
```

192.0.2.0 是可以允许连接的子网地址

客户端机器上，编辑/etc/chrony.conf

```
server master
driftfile /var/lib/chrony/drift
logdir /var/log/chrony
log measurements statistics tracking
keyfile /etc/chrony.keys
commandkey 24
local stratum 10
initstepslew 20 master
allow 192.0.2.123
```

192.0.2.123 是 master 主机的地址。

4.4.5. 使用 chronyc

4.4.5.1. 使用 chronyc 控制 chronyd

进入 chronyc 的交互界面

```
~]# chronyc -a
```

chronyc 必须以 root 用户执行。-a 选项是使用本地 keys 自动登录

chronyc 命令行界面

```
chronyc>
```

您可以使用 help 显示所有命令

也可以使用非交互界面进行输入

```
chronyc command
```

4.4.5.2. 使用 chronyc 进行远程管理

配置 chrony 连接到远程 chronyd，格式如下：

```
~]$ chronyc -h hostname
```

指定端口

```
~]$ chronyc -h hostname -p port
```

port 是用来控制和监控远程 chronyd

4.5. 配置 NTP 使用 NTPD

4.5.1. NTP 介绍

NTP 是网络时间协议(Network Time Protocol)，它是用来同步网络中各个计算机的时间的协议。

4.5.2. NTP 分层

NTP 分层如下

Stratum 0:

原子钟和信号广播电台和 GPS

- GPS (Global Positioning System)
- Mobile Phone Systems
- Low Frequency Radio Broadcasts WWVB (Colorado, USA.), JJY-40 and JJ Y-60 (Japan), DCF77 (Germany), and MSF (United Kingdom)

这些信号可以被专用的设备接收，并经常被 RS-232 连接到系统作为一个组织或站点范围内的时间服务器。

Stratum 1：

电脑附加了无线时钟、GPS 时钟或原子钟

Stratum 2：

从 Stratum 1 读取，作为更底层的服务器

Stratum 3：

从 Stratum 2 读取，作为更底层的服务器

Stratum n+1：

从 Stratum n 读取，作为更底层的服务器

Stratum 15：

从 Stratum 14 读取，作为更底层的服务器

4.5.3. 理解 NTP

中标麒麟高级服务器操作系统软件使用的 NTP 版本，在 RFC 1305 Networ

k Time Protocol (Version 3) Specification, Implementation and Analysis 和 RFC 5905 Network Time Protocol Version 4: Protocol 和 Algorithms Specification 有详细描述。

4.5.4. 理解 drift 文件

drift 文件记录保存着系统时间频率与 UTC 时钟源频率的偏移量。

4.5.5. UTC，TIMEZONES 和 DST

NTP 完全在 UTC (Universal Time, Coordinated)中，Timezones 和 DST (Daylight Saving Time) 被本地系统应用。/etc/localtime 是/usr/share/zoneinfo 的拷贝或链接。RTC 可能在本地时间或者 UTC 中，在/etc/adjtime 第三行指定。这个文件可以知道 RTC 怎么被设置。用户可以很方便的使用 System Clock Uses UTC 在 Date and Time 图形配置界面进行配置的修改。

4.5.6. NTP 身份验证选项

在当前网络，攻击着可以通过发送 NTP 包进行攻击。在使用公共的 NTP 源时，为了减少风险，在/etc/ntp.conf 文件中，必须有 3 个公共源。假如只有一个源，ntpd 将会忽略该源。根据应用的风险需要，可以选择开启或不开启身份验证。

默认的组播和广播是需要验证的。假如您决定关闭身份验证，可以使用 disable auth 在 ntp.conf 文件中。

4.5.7. 在虚拟机中管理时间

虚拟机不能使用真实的 hardware clock 并且虚拟机时间不够稳定。在中标麒麟高级服务器操作系统软件中，kvm 默认使用 kvm-clock。

4.5.8. 理解闰秒

闰秒，是指为保持协调世界时接近于世界时时刻，由国际计量局统一规定在年底或年中（也可能在季末）对协调世界时增加或减少 1 秒的调整。由于地球自转的不均匀性和长期变慢性（主要由潮汐摩擦引起的），会使世界时（民用时）和原子时之间相差超过到±0.9 秒时，就把世界时向前拨 1 秒（负闰秒，最后一分钟为 59 秒）或向后拨 1 秒（正闰秒，最后一分钟为 61 秒）； 闰秒一般加在公历年末或公历六月末。

4.5.9. 理解 ntpd 配置文件

守护进程 ntpd 在系统启动或重启时读取配置文件/etc/ntp.conf，您可以通过下面命令查看配置文件。

```
~]$ less /etc/ntp.conf
```

下面介绍默认的配置项

The driftfile entry

drift 文件路径

```
driftfile /var/lib/ntp/drift
```

The access control entries

以下行设置默认访问控制限制:

```
restrict default nomodify notrap nopeer noquery
```

nomodify 选项：阻止修改配置文件

notrap 选项：防止 ntpdc 控制消息协议陷阱

nopeer 选项：防止 peer 联合的形成

noquery 选项：防止 ntpq 和 ntpdc 查询,但没有时间查询

有时候各种进程和应用需要 127.0.0.0/8 地址段。默认 restrict 行阻止所有非允许的接入

```
# the administrative functions.
restrict 127.0.0.1
restrict ::1
```

如果特别需要被另一个应用程序，可以在下面添加地址。

主机在本地网络是不允许的，因为默认的 restrict。为了进行修改，例如，允许 192.0.2.0/24 网络进行时间的查询

```
restrict 192.0.2.0 mask 255.255.255.0 nomodify notrap nopeer
```

如果只是允许某一台主机，例如 192.0.2.250/32

```
restrict 192.0.2.250
```

如果没有指定 mask，255.255.255.255 将会被指定

The public servers entr

默认的，ntp.conf 配置文件包含四个公共服务器

```
server 0.rhel.pool.ntp.org iburst
```

```
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
```

The broadcast multicast servers entry

默认的，ntp.conf 包含了很多被注释掉的例子，这些在很大程度上是自我解释。

4.5.10. 理解 ntpd 的 sysconfig 文件

这个文件将会在 ntpd 启动服务初始化脚本的时候读取。默认的内容如下：

```
# Command line options for ntpd
OPTIONS="-g"
```

-g 选项可以使 ntpd 忽略 1000s 的偏移量限制，尝试同步时间即使偏移量大于 1000s，但是仅仅在系统启动的时候。当离开这个选项的时候，ntpd 在偏移量大于 1000s 的时候自动退出。

4.5.11. 禁止 chrony

命令如下

```
~]# systemctl stop chronyd
```

取消 chrony 开机启动选项

```
~]# systemctl disable chronyd
```

查看状态

```
~]$ systemctl status chronyd
```

4.5.12. 检查 NTP 守护进程是否安装

命令如下

```
~]# yum install ntp
```

4.5.13. ntpd 的安装

```
~]# yum install ntp
```

设置 ntpd 开机启动

```
~]# systemctl enable ntpd
```

4.5.14. 检查 ntp 的状态

检查 ntpd 是否运行

```
~]$ systemctl status ntpd
```

获取 ntpd 的状态报告

```
~]$ ntpstat
```

4.5.15. 配置防火墙允许 ntp 包进入

ntp 使用 UDP 包，端口 123 。必须要能够允许通过防火墙。

检查防火墙时候允许 ntp 传输，使用图形界面 **Firewall Configuration** 工具。

启动 **firewall-config**，点击 **Super** 键进入，输入 firewall 并且按回车键，防火墙配置窗口被打开。输入用户密码。

命令行进入

```
~]# firewall-config
```

寻找“连接”一词在左下角。这表明 firewall-config 工具连接到用户空间守护进程,firewalld。

4.5.15.1. 修改 firewall 配置

为了能够让配置生效，确保下拉菜单 **Configuration** 是设置为 Runtime。或者修改配置文件在下一次启动生效，在下拉菜单中选择 Permanent

4.5.15.2. 打开防火墙端口

打开 firewall-config 工具，选择网络，选择端口标签，点击“添加”按钮。Port and Protocol 窗口被打开，输入端口 123，下拉菜单选择 udp

4.5.16. 配置 ntpdate 服务器

ntpdate 服务是为了设置时间在系统启动的时候。

检查 ntpdate 服务是否运行

```
~]$ systemctl status ntpdate
```

设置开机启动

```
~]# systemctl enable ntpdate
```

在中标麒麟高级服务器操作系统软件系列，默认/etc/ntp/step-tickers 文件包含 0.rhel.pool.ntp.org。添加额外的 ntpdate 服务器，编辑/etc/ntp/step-tickers。服务器的数量是不重要的，因为 ntpdate 只是使用这个获取一次时间信息。假如您有一个外网的时间服务器，在第一行输入该服务器的地址。在第二行输入备份的服务器。

4.5.17. 配置 ntp

修改默认的 ntp 服务器配置，编辑/etc/ntp.conf 文件。该文件是和 ntpd 一起被安装。

4.5.17.1. 配置 NTP 服务访问控制

编辑 ntp.conf，使用 restrict 命令

```
# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

restrict 使用

```
restrict option
```

option：可以是一个和多个

- ignore——所有的数据包将被忽略,包括 ntpq 和 ntpdc 查询
- kod——一个“Kiss-o'-death”包发送以减少不必要的查询
- limited——如果数据包违反了速率限制，不响应服务请求。ntpq 和 ntpdc 查询不受影响
- lowpriotrap——低优先级匹配主机设定的陷阱
- nomodify——阻止修改配置
- noquery——阻止 ntpq 和 ntpdc 查询，不包括时间查询
- nopeer——阻止 peer 联合形成
- noserve——除了 ntpq 和 ntpdc 查询，拒绝所有的包
- notrap——防止 ntpdc 控制消息协议陷阱
- notrust——拒绝没有密码身份验证数据包

- ntpport——修改算法，只匹配 NTP UDP port123
- version——拒绝不匹配当前 NTP 版本的数据包

为了不相应所有的查询，配置速率限制，restrict 可以使用 limited 选项。假如 ntpd 必须响应 KoD 数据包，restrict 必须使用 limited 和 kod 选项。

ntpq 和 ntpdc 查询可用于放大的攻击，不要移除 noquery 选项。

4.5.17.2. 配置连接 NTP 服务器的速率限制

给 restrict 加上 limited 选项，如果您不想使用默认放弃的参数，可以使用 discard 命令。

命令格式如下

```
discard [average value] [minimum value] [monitor value]
```

- average——指定允许的最小平均数据包间隔,它接受一个参数 log2 秒。默认值是 3（2 的三次方，相当于 8）
- minimum——指定允许的最低包间距,在 log2 秒它接受一个参数。默认值是 1(2 的一次方，相当于 2 秒)
- monitor——指定数据包的丢弃概率，一旦允许利率已经超过限制。默认值为 3000 秒。这个选项适用于服务器每秒获得 1000 或更多请求

discard 命令例子

```
discard average 4
```

```
discard average 4 minimum 2
```

4.5.17.3. 添加 peer 地址

在 ntp.conf 配置文件添加 peer 命令

```
peer address
```

address 是一个 ip 地址或 DNS 可识别的名称。address 必须是在同一层的系统。Peers 必须拥有至少一个不同的时间源。

4.5.17.4. 添加服务器地址

添加一个服务器地址，该服务器是运行 NTP 服务且处在更高层。在 ntp.conf 文件，使用 server 命令

```
server address
```

address 是可识别的 IP 地址或 DNS 识别的名称。

4.5.17.5. 添加一个广播或多播服务器地址

添加一个广播或多播发送地址，也就是说，广播或多播 NTP 包到达的地址。在 ntp.conf 文件中，使用 broadcast

使用如下

```
broadcast address
```

4.5.17.6. 添加一个 Manycast 客户地址

添加 manycast 客户端地址，在 ntp.conf 文件中，添加 manycastclient 命令。

格式如下

```
manycastclient address
```

该命令配置系统作为一个 NTP 客户端。系统可以同时为客户端和服务器端。

4.5.17.7. 添加 Broadcast 客户端地址

添加 broadcast 客户端地址，在 ntp.conf 文件中，添加 broadcastclient 命令。

格式如下

```
broadcastclient
```

该命令配置系统作为一个 NTP 客户端。系统可以同时为客户端和服务器端。

4.5.17.8. 添加一个 Manycast 服务器地址

添加 manycast 服务器地址，在 ntp.conf 文件中，添加 manycastserver 命令

格式如下

```
manycastserver address
```

该命令配置系统作为一个 NTP 服务器。系统可以同时为客户端和服务器端。

4.5.17.9. 添加一个 Multicast 服务器地址

添加 multicast 服务器地址，在 ntp.conf 文件中，添加 multicastclient 命令

格式如下

```
multicastclient address
```

该命令配置系统作为一个 NTP 服务器。系统可以同时为客户端和服务器端。

4.5.17.10. 配置 Burst 选项

不要和公共 NTP 服务器一起使用该选项，该选项适用于在自己组织里的应用程序。

在服务器命令结尾添加该选项

```
burst
```

4.5.17.11. 配置 iburst 选项

在服务器命令结尾添加该选项

```
iburst
```

4.5.17.12. 使用 key 配置 Symmetric Authentication

在服务器或 peer 命令后，添加该选项

```
key number
```

number 是一个 1 到 65534 的数。该选项可以在数据包中使用 message authentication code (MAC)。该选项和 peer，server，broadcast，和 manycastclient 命令使用。

/etc/ntp.conf，格式如下：

```
server 192.168.1.1 key 10
broadcast 192.168.1.255 key 20
manycastclient 239.255.254.254 key 30
```

4.5.17.13. 配置 Poll Interval

修改默认的 poll interval，在服务器或 peer 命令后，添加该选项

```
minpoll value and maxpoll value
```

4.5.17.14. 配置服务器优先级

指定一个高优先级的服务器，在 server 或 peer 命令后添加该选项

```
prefer
```

4.5.17.15. 为 NTP 数据包配置 Time-to-Live

在 server 或 peer 命令后添加该选项

```
ttl  value
```

4.5.17.16. 配置 NTP 使用版本

在 server 或 peer 命令后添加该选项

```
version  value
```

4.5.18. 配置硬件时钟更新

配置系统时间更新硬件时钟（RTC)，在/etc/sysconfig/ntpdate 添加以下选项

```
SYNC_HWCLOCK=yes
```

命令使用如下

```
~]#  hwclock  --systohc
```

4.5.19. 配置时钟源

在系统列出可用的时钟源

```
~]$  cd  /sys/devices/system/clocksource/clocksource0/clocksource0]
$  cat  available_clocksource
$  cat  current_clocksource
```

覆盖默认的时钟源，在内核的 GRUB 里添加 clocksource，例如

```
~]#  grubby  --args=clocksource=tsc  --update-kernel=DEFAULT
```

4.6. 使用 ptp4l 配置 PTP

4.6.1. PTP 介绍

Precision Time Protocol (PTP) 是用于网络中时钟同步的协议。使用时结合

硬件支持，要比普通的 NTP 更快。PTP 的支持被划分为内核和用户空间。中标麒麟高级服务器操作系统软件 linux 内核支持 PTP 时钟。linuxptp 是该协议的实现。

这个 linuxptp 包包含 ptp4l 和 phc2sys 时钟同步项目。ptp4l 程序实现了 PTP 边界时钟和普通时钟。与硬件时间戳,它用于 PTP 硬件时钟与主时钟同步，与软件时间戳，它用于系统时钟与主时钟同步。phc2sys 程序只需与硬件时间戳，将系统时钟同步到 PTP 硬件时钟网络接口卡(NIC)。

4.6.1.1. PTP 的优点

和 Network Time Protocol (NTP)相比，PTP 最主要的优点是硬件的支持，包括许多的 network interface controllers (NIC)和 network switches。极大的提高了时间同步的准确性。

4.6.2. 使用 PTP

为了使用 PTP，内核网络驱动必须支持软件或硬件时间戳。

4.6.2.1. 检查驱动和硬件支持

使用 ethtool 命令查询

```
~]# ethtool -T eth3          o
```

eth3 是您想检查的接口

如果支持软时间戳，必须包含以下参数

- SOF_TIMESTAMPING_SOFTWARE
- SOF_TIMESTAMPING_TX_SOFTWARE
- SOF_TIMESTAMPING_RX_SOFTWARE

如果支持硬件时间戳，必须包含以下参数

- SOF_TIMESTAMPING_RAW_HARDWARE
- SOF_TIMESTAMPING_TX_HARDWARE
- SOF_TIMESTAMPING_RX_HARDWARE

4.6.2.2. 安装 PTP

```
~]# yum install linuxptp
```

这将会安装 ptp4l 和 phc2sys。

4.6.2.3. 启动 ptp4l

ptp4l 可以通过命令行或者作为服务启动。当 ptp4l 作为一个服务时，可以在/etc/sysconfig/ptp4l 指定选项。不管是作为服务或是命令行启动，必须在/etc/ptp4l.conf 指定选项。/etc/sysconfig/ptp4l 文件包含-f /etc/ptp4l.conf 行，这会让 ptp4l

读取/etc/ptp4l.conf 文件。

作为服务启动 ptp4l

```
~]# systemctl start ptp4l
```

使用命令行运行 ptp4l

```
~]# ptp4l -i eth3 -m
```

输出结果如下

```
~]# ptp4l -i eth3 -m
```

记录 ptp4l 日志

默认情况下，日志文件是发送到/var/log/messages。-m 选项打开日志的标准输出，能够为 debugging 提供信息。

打开软时间戳，-s 选项，使用如下

```
~]# ptp4l -i eth3 -m -S
```

选择一个延迟测量机制

有 2 种不同的延迟测量机制，能够通过不同的选项进行选择：

-P

-P 选择 peer-to-peer (P2P) 延迟测量机制

P2P 机制是首选，因为它对网络拓扑的变化更快，可能比其他机制更准确测量延迟。

-E

-E 选择 end-to-end (E2E)延迟测量机制。这是默认的选项

E2E 也被认为是 request-response 延迟机制

-A

-A 打开延迟机制的自动选择

自动选项打开的是 ptp4l 的 E2E 模式。如有需要，可以改成 P2P 模式。

4.6.3. 和多个接口使用 PTP

在不同网络的多接口中使用 PTP，有必要改变反向路径转发模式为松散模式。

sysctl 组件用来对内核进行读写。对正在运行的系统，可以使用命令行或者是修改/etc/sysctl.conf 文件

修改为 loose 模式，命令如下

```
~]# sysctl -w net.ipv4.conf.default.rp_filter=2
sysctl -w net.ipv4.conf.all.rp_filter=2
```

为了对每一个接口修改反向路径转发模式，使用 net.ipv4.interface.rp_filter 命令行，例如，em1 网络接口

```
~]# sysctl -w net.ipv4.conf.em1.rp_filter=2
```

为了让修改的配置永久有效，修改/etc/sysctl.conf 文件。例如：为了修改所有的网卡的模式，修改/etc/sysctl.conf，如下所示

```
net.ipv4.conf.all.rp_filter=2
```

如果是修改单一的某个网卡模式，如下所示：

```
net.ipv4.conf.interface.rp_filter=2
```

4.6.4. 指定一个配置文件

没有默认的配置文件，所以需要在运行的时候指定，使用-f 选项

```
~]# ptp4l -f /etc/ptp4l.conf
```

一个配置文件内容，相当于-i eth3 -m –S 选项，如下所示

```
~]# cat /etc/ptp4l.conf
```

4.6.5. 使用 PTP 管理客户端

PTP 的管理客户端，pmc 可以用来获取额外 ptp4l 信息。

```
~]# pmc -u -b 0 'GET CURRENT_DATA_SET'
```

```
~]# pmc -u -b 0 'GET TIME_STATUS_NP'
```

查看 pmc 全部命令

```
~]# pmc help
```

4.6.6. 同步时钟

phc2sys 程序用来将系统时间同步到 PTP 的 hardware clock (PHC)。phc2sys 服务配置文件/etc/sysconfig/phc2sys。默认配置如下：

```
OPTIONS="-a -r"
```

-a 选项使得 phc2sys 能够同步来自 ptp4l 应用的时钟。它将跟随 PTP 端口状态的变化，相应调整网卡的硬件之间的同步时钟，系统时钟将不会被同步，除非-r 选项被指定。如果您想让系统时钟成为一个源，指定-r 选项两次。

修改/etc/sysconfig/phc2sys 后，重启 phc2sys 服务

```
~]# systemctl restart phc2sys
```

正常情况下，使用 systemctl 命令来启动，停止和重启 phc2sys 服务。

如果您不想使用服务启动 phc2sys，您可以通过命令行来启动。

```
~]# phc2sys -a -r
```

-a 选项使得 phc2sys 能够同步来自 ptp4l 应用的时钟。如果您想让系统时钟成为一个源，指定-r 选项两次。

使用-s 选项同步系统时钟至特定的 PTP 硬件时钟，例如：

```
~]# phc2sys -s eth3 -w
```

-w 选项等待运行的 ptp4l 应用同步 PTP 时钟，恢复 TAI 至 UTC 偏移量

正常情况下，PTP 操作在 nternational Atomic Time (TAI)，系统时间保持在 Coordinated Universal Time (UTC)。当前 TAI 和 UTC 的偏移量是 35s。当闰秒插入或删除的时候，偏移量会进行改变，这个变化每几年会发生一次。当-w 选项没有使用时，可以使用-O 选项来进行偏移量的设置

```
~]# phc2sys -s eth3 -O -35
```

当 phc2sys servo 处于锁状态时，时钟将不会走，除非-S 选项能够被使用。这意味着 phc2sys 程序必须在 ptp4l 程序以及和 PTP 硬件时钟同步后启动。使用-w 选项，phc2sys 不用在 ptp4l 后启动，因为它会等待，直到 ptp4l 已经同步。

phc2sys 可以作为服务启动

```
~]# systemctl start phc2sys
```

当以服务进行启动，可以将选项在/etc/sysconfig/phc2sys 文件指定。

4.6.7. 验证时间同步

当 PTP 时间同步工作正常的时候，新的频率偏移量以及调整消息将会打印 ptp4l 和 phc2sys。这些信息可以在 /var/log/messages 文件查看。

```
ptp4l[352.359]: selected /dev/ptp0 as PTP clock
ptp4l[352.361]: port 1: INITIALIZING to LISTENING on INITIAL
IZE
ptp4l[352.361]: port 0: INITIALIZING to LISTENING on INITIAL
IZE
……
```

phc2sys 输出的例子

```
phc2sys[526.527]: Waiting for ptp4l...
phc2sys[527.528]: Waiting for ptp4l...
phc2sys[528.528]: phc offset 55341 s0 freq +0 delay 2
729
phc2sys[529.528]: phc offset 54658 s1 freq -37690 delay 2
725
……
```

ptp4l 有一个命令 summary_interval，可以减少输出，默认情况下，每一秒会打印一条信息。可以修改为每 1024s 打印一次，在/etc/ptp4l.conf 文件添加以下行：

```
summary_interval 10
```

一个 ptp4l 输出的例子，使用 summary_interval 6

```
ptp4l: [615.253] selected /dev/ptp0 as PTP clock
ptp4l: [615.255] port 1: INITIALIZING to LISTENING on INITIA
LIZE
……
```

减少从 phc2sys 的输出，可以使用-u 选项

```
~]# phc2sys -u summary-updates
```

summary-updates 是时钟更新数，例子如下：

```
~]# phc2sys -s eth3 -w -m -u 60
```

4.6.8. 使用 NTP 服务 PTP 时间

ntpd 守护进程可以配置从系统时间分发时间，系统时间是由 ptp4l 和 phc2sy

s 使用 LOCAl 时间驱动进行同步。为了防止 ntpd 调整系统时钟，ntp.conf 文件必须不能够指定任何 NTP 服务器。以下是一个例子：

```
~]# cat /etc/ntp.conf
server 127.127.1.0
fudge 127.127.1.0 stratum 0
```

4.6.9. 使用 PTP 服务 NTP 时间

NTP 至 PTP 同步也是可能的。当 ntpd 是用来同步系统时间，ptp4l 可以使用 priority1 选项配置为 grandmaster 时钟，通过 PTP 系统时钟来分发时间。

```
~]# cat /etc/ptp4l.conf
[global]
priority1 127
[eth3]
# ptp4l -f /etc/ptp4l.conf
```

使用硬件时间戳，需要使用 phc2sys 来同步 PTP 硬件时钟至系统时钟。假如 phc2sys 是一个服务，编辑/etc/sysconfig/phc2sys 文件，默认的配置如下

```
OPTIONS="-a -r"
```

使用 root 用户，编辑

```
~]# vi /etc/sysconfig/phc2sys
OPTIONS="-a -r -r"
```

在这里，-r 选项被指定 2 次，允许 PTP 网卡硬件时钟从系统时钟进行同步。重启 phc2sys

```
~]# systemctl restart phc2sys
```

防止 PTP 时钟频率快速的变化，可以通过设置更小的 P (proportional) 和 I (integral)

```
~]# phc2sys -a -r -r -P 0.01 -I 0.0001
```

4.6.10. 使用 timemaster 同步 PTP 或 NTP 时间

可以使用 timemaster 程序让系统时间跟可用的时间源进行同步。PTP 时间是由 phc2sys 和 ptp4l 提供，通过使用 shared memory driver。NTP 守护进程能够

比较所有的源，选择最优的源进行同步。

启动时，timemaster 会读取一个配置文件，该配置文件会指定 NTP 和 PTP 时间源，检查哪些网络接口有自己或共享的 PTP 硬件时钟，生成 ptp4l 和 chrony d 或 ntpd 配置文件，启动 ptp4l，phc2sys 和 phc2sys 或 ntpd。

4.6.10.1. 作为一个服务启动 timemaster

使用 root 用户

```
~]# systemctl start timemaster
```

启动时，会读取/etc/timemaster.conf 配置文件

4.6.10.2. 理解 timemaster 配置文件

默认的配置文件如下所示：

```
~]$ less /etc/timemaster.conf
# Configuration file for timemaster

#[ntp_server ntp-server.local]
#minpoll 4
#maxpoll 4

#[ptp_domain 0]
#interfaces eth0

[timemaster]
ntp_program chronyd

[chrony.conf]
include /etc/chrony.conf

[ntp.conf]
includefile /etc/ntp.conf

[ptp4l.conf]

[chronyd]
```

```
path  /usr/sbin/chronyd
options  -u  chrony

[ntpd]
path  /usr/sbin/ntpd
options  -u  ntp:ntp  -g

[phc2sys]
path  /usr/sbin/phc2sys

[ptp4l]
path  /usr/sbin/ptp4l
```

注意到部分命名如下

```
[ntp_server  address]
```

这是一个 NTP 服务器区域的例子，ntp-server.local 是一个本地 LAN 的 NTP 服务器。

注意到部分命名如下

```
[ptp_domain  number]
```

PTP domain 是一组 PTP 时钟，它们能够相互同步。它们可以或者不可以和其它域进行同步。拥有相同域的数字组成一个域。这包括一个 PTP 的 grandmaster 时钟。

注意到部分命名如下

```
[timemaster]
```

默认的 timemaster 配置文件包含了 ntpd 和 chrony 配置（/etc/ntp.conf 或者/etc/chronyd.conf），为了能够包含访问限制的配置和认证密钥。这意味着任何指定的 NTP 服务器能够和 timemaster 一起被使用。

4.6.10.3.配置 timemaster 选项

实例：编辑 timemaster 配置文件

1) 打开/etc/timemaster.conf 配置文件。
2) 对于每一个 NTP 服务器，您想控制使用 timemaster，创建[ntp_server address]字段。

3) 添加要在域中使用的网卡，编辑#[ptp_domain 0]字段并添加网卡，例子如下：

```
[ptp_domain 0]
        interfaces eth0

        [ptp_domain 1]
        interfaces eth1
```

4) 假如需要使用 ntpd 作为 NTP 守护进程，在[timemaster]字段修改默认的 entry，chronyd 改为 ntpd。
5) 假如使用 chronyd 作为 NTP 服务器，在[chrony.conf]字段添加额外的选项，在默认的 include /etc/chrony.conf entry 下。
6) 假如使用 ntpd 作为 NTP 服务器，在[chrony.conf]字段添加额外的选项，在默认的 include /etc/ntp.conf entry 下。
7) 在[ptp4l.conf]字段，添加任何将要拷贝到 ptp4l 生成的配置文件的选项。
8) 在[chronyd]字段，添加任何命令行，当被 timemaster 访问时该命令需要传递至 chronyd。
9) 在[ntpd]字段，添加任何命令行，当被 timemaster 访问时该命令需要传递至 ntpd。
10) 在[phc2sys]字段，添加任何命令行，当被 timemaster 访问时该命令需要传递至 phc2sys。
11) 在[ptp4l]字段，添加任何命令行，当被 timemaster 访问时该命令需要传递至 ptp4l。
12) 保存配置文件，重启 timemaster。

```
~]# systemctl restart timemaster
```

4.6.11. 提高准确性

ptp4l 和 phc2sys 应用能够配置为使用新的 adaptive servo。对于 PI servo 来说，它的优势在于不需要配置 PI 优化配置文件。在 ptp4l 中使用，需要在/etc/ptp4l.conf 配置文件添加以下行：

```
clock_servo linreg
```

重启 ptp4l 服务

```
~]# systemctl restart ptp4l
```

在 phc2sys 中使用，需要在/etc/sysconfig/phc2sys 配置文件添加以下行：

```
-E linreg
```

重启 phc2sys 服务

```
~]# systemctl restart phc2sys
```

第五章 KVM 虚拟化

5.1. 安装与配置

5.1.1. 环境要求

安装中标麒麟高级服务器操作系统软件 V7.0 时，在软件选择时勾选虚拟机相关选项。

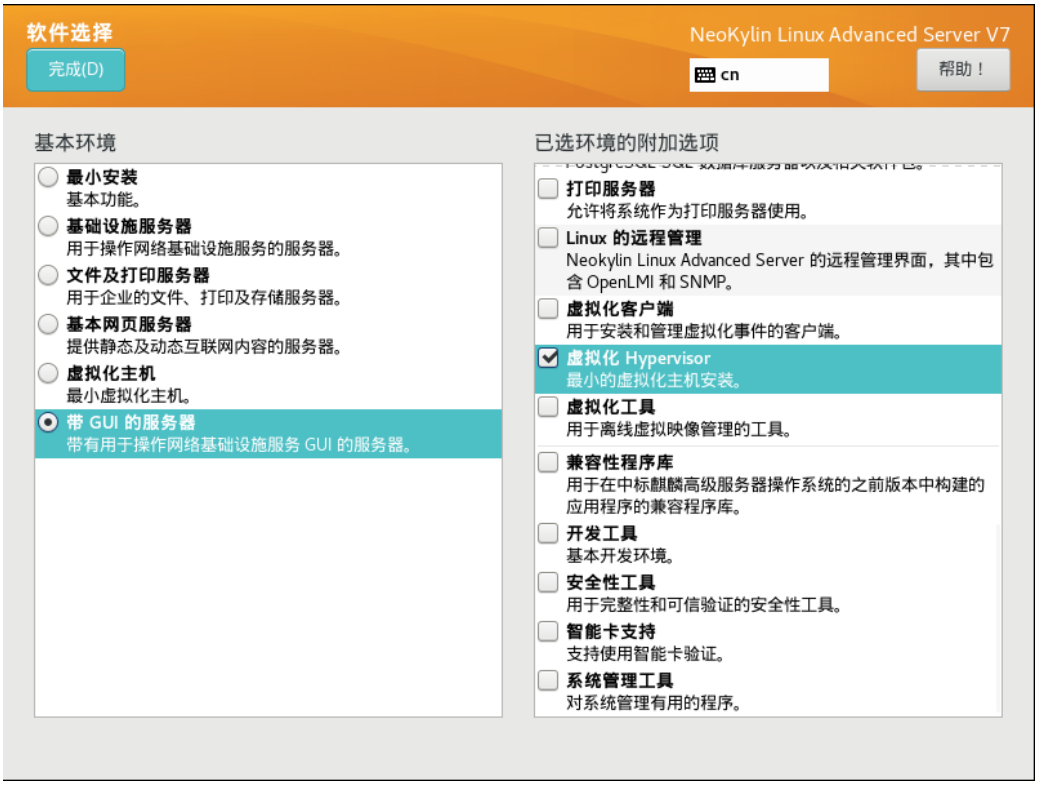


图 5-1 安装虚拟化软件包选择

5.1.2. 开启虚拟化服务

系统安装成功之后，进系统开启虚拟化服务（开启此服务时需要使用 root 用户权限）。

```
systemctl start libvirtd
```

```
systemctl enable libvirtd
```

5.2. 虚拟机的使用

5.2.1. 启动虚拟机

您可以点击应用程序-系统工具-虚拟系统管理器，也可以在终端输入 virt-manager 来启动虚拟机。

5.2.2. 本地介质安装

5.2.2.1. 新建虚拟机

点击**文件--新建虚拟机**，或者点击如下图中的按钮即可新建虚拟机。

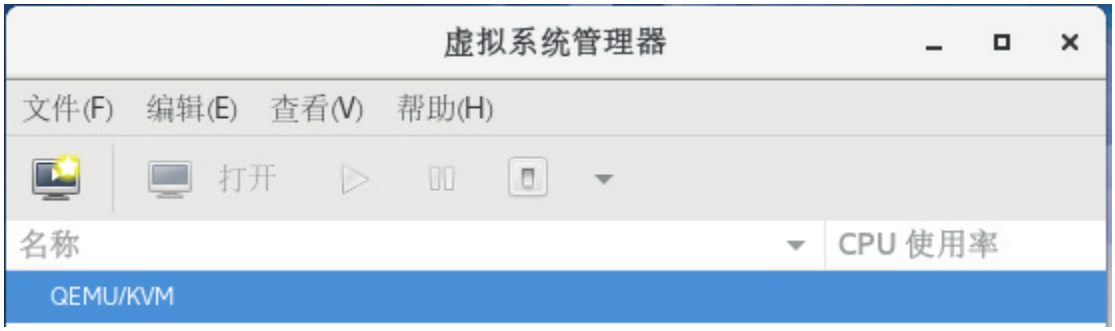


图 5-2 新建虚拟机

5.2.2.2. 本地介质安装

选择安装系统的方式，常用的方式是【本地安装介质】，选择虚拟的机器架构类型。

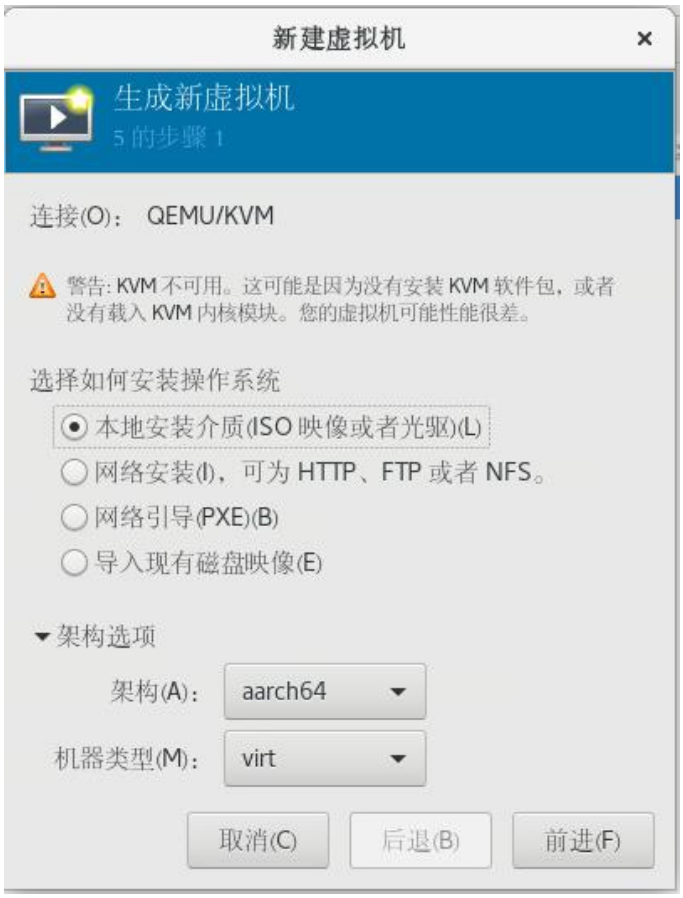


图 5-3 新建虚拟机

5.2.2.3. 选择 ISO 镜像

添加本地 ISO 镜像文件，并取消选中“根据安装介质自动侦测操作系统”，操作系统类型选择“Linux”，版本根据要安装的系统进行选择。

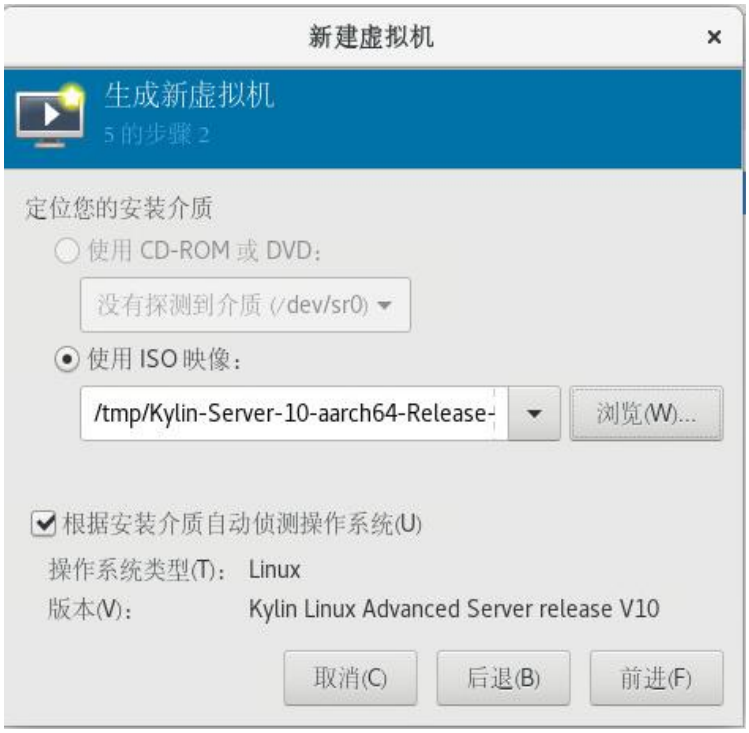


图 5-4 选择 IOS 镜像

5.2.2.4. 选择内存和 CPU 设置

根据实体机情况以及虚拟机实际使用情况，来选择内存大小和 CPU 核数。

图 5-5 选择内存和 CPU 设置

5.2.2.5. 存储设置

根据需求选择虚拟磁盘镜像的大小，可以不使用默认的磁盘镜像位置，在有较大空间的位置，使用命令创建虚拟磁盘镜像，在此步中，就可以使用第二项【选择或创建自定义存储】，选择自己创建的镜像文件。

创建虚拟磁盘镜像命令：qemu-img create -f qcow2 xxx.qcow2 xG

(-f 为镜像类型，多数情况选择 qcow2 格式；xxx.qcow2 为镜像名称，可自行修改；xG 为镜像大小，可根据实际情况设定)

图 5-6 存储设置

5.2.2.6. 虚拟机命名和网络配置

此步可以对虚拟机进行命名，来区分不同虚拟机，还提供了默认的网络配置“NAT”。并且勾选【在安装前自定义配置】

图 5-7 虚拟机的命名和网络配置

5.2.2.7. 安装前自定义配置

选择磁盘总线：选中 **SCSI CDROM1**，点击高级选项，磁盘总线选择 SCSI。

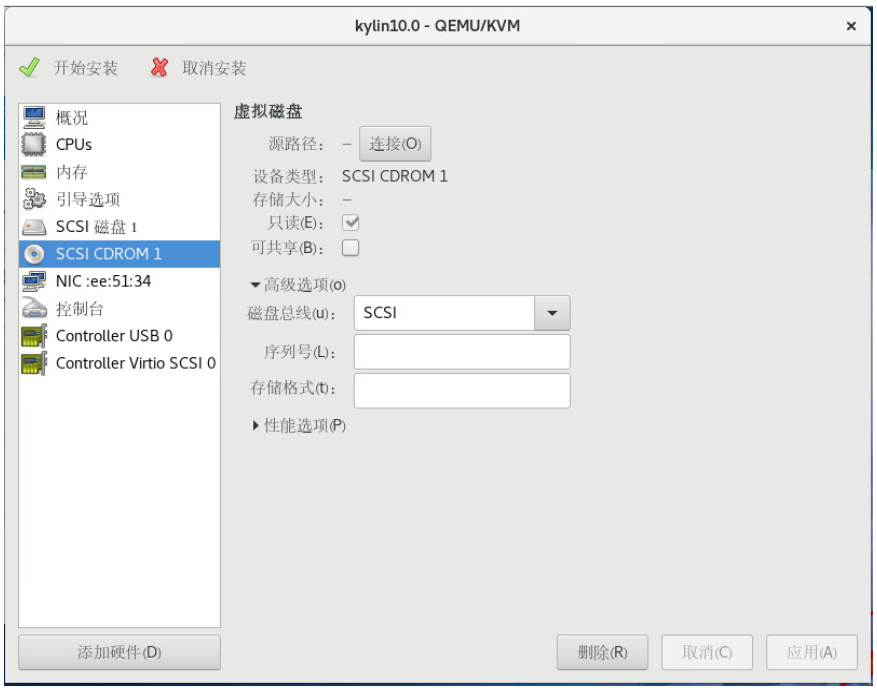


图 5-8 选择磁盘总线

点击**添加硬件**，即可添加硬件。

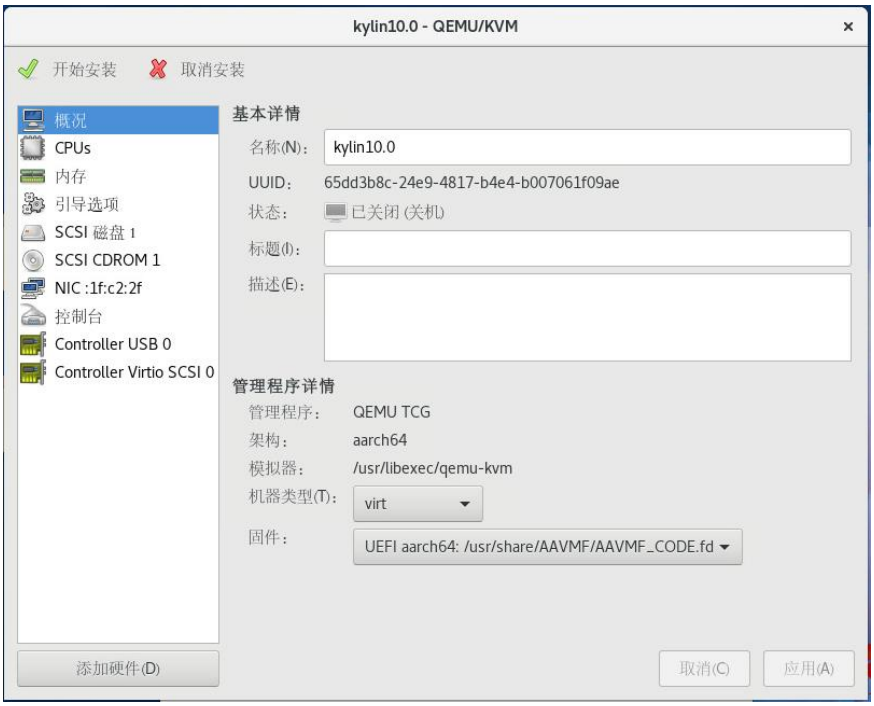


图 5-9 硬件添加

添加**图形**直接添加即可，**视频**选择 Virtio，这两个硬件提供图形界面，方便使用，之后点击**完成**。

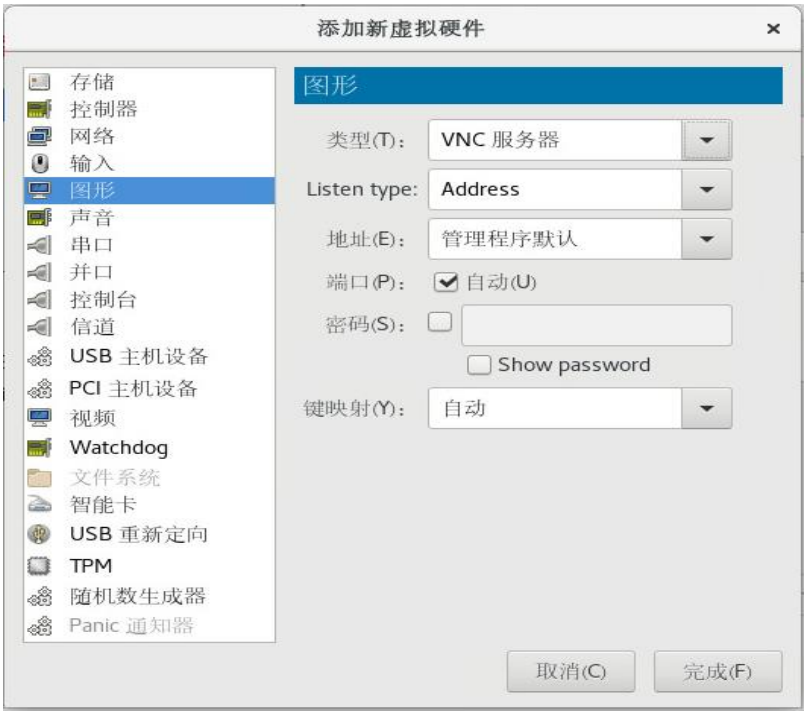


图 5-10 添加图形，视频及输入（1）

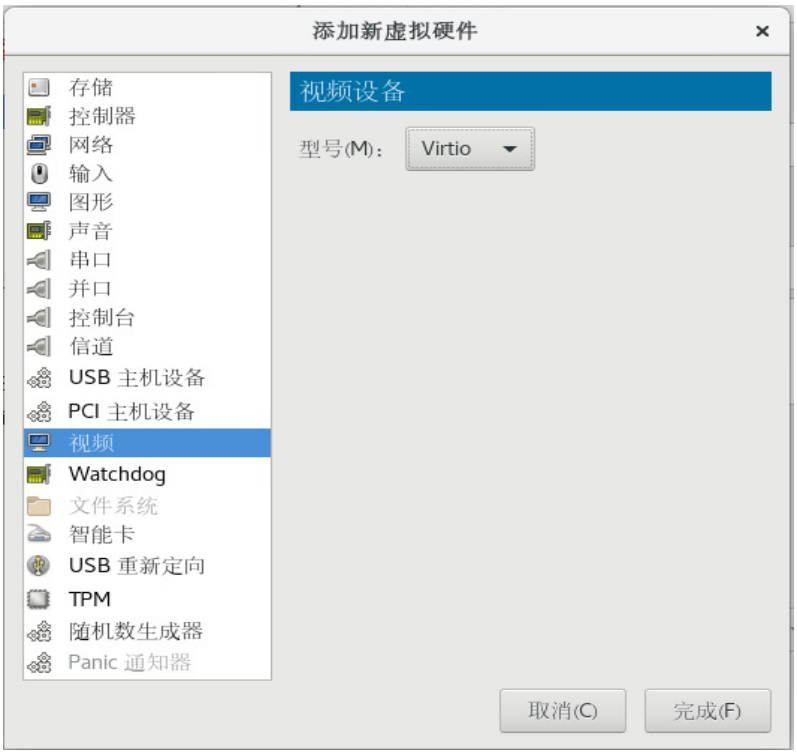


图 5-10 添加图形，视频及输入（2）

图 5-10 添加图形，视频及输入（3）

5.2.2.8. 安装系统

安装系统与正常系统安装步骤一致（系统安装详见《中标麒麟高级服务器操作系统软件安装手册》）。

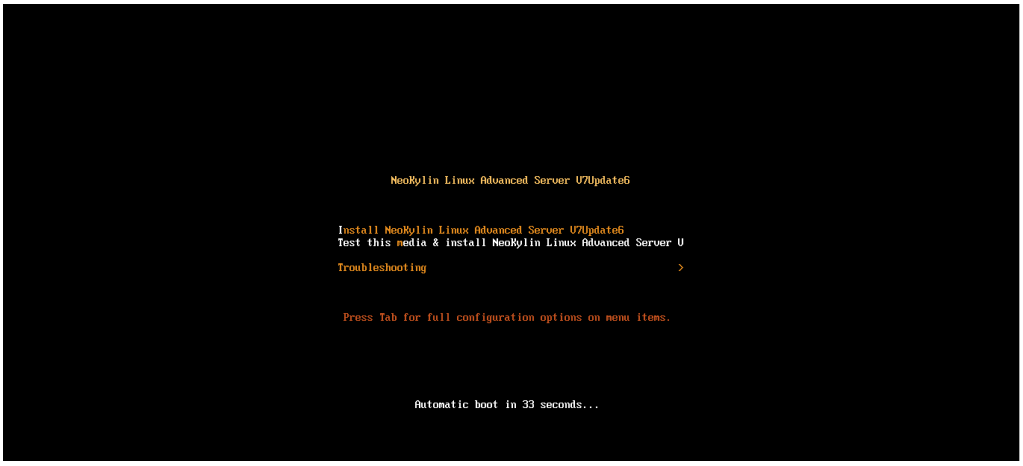


图 5-11 安装启动

5.2.3. Qcow2 导入现有磁盘镜像

使用现有的磁盘镜像生成虚拟机，需要提前准备好磁盘镜像。

1. 新建虚拟机，选择导入现有磁盘映像

图 5-12 导入现有磁盘映像

2. 选择本地的磁盘镜像(磁盘镜像的路径不能是 home 下的用户目录下，否则会出现没有权限的问题)

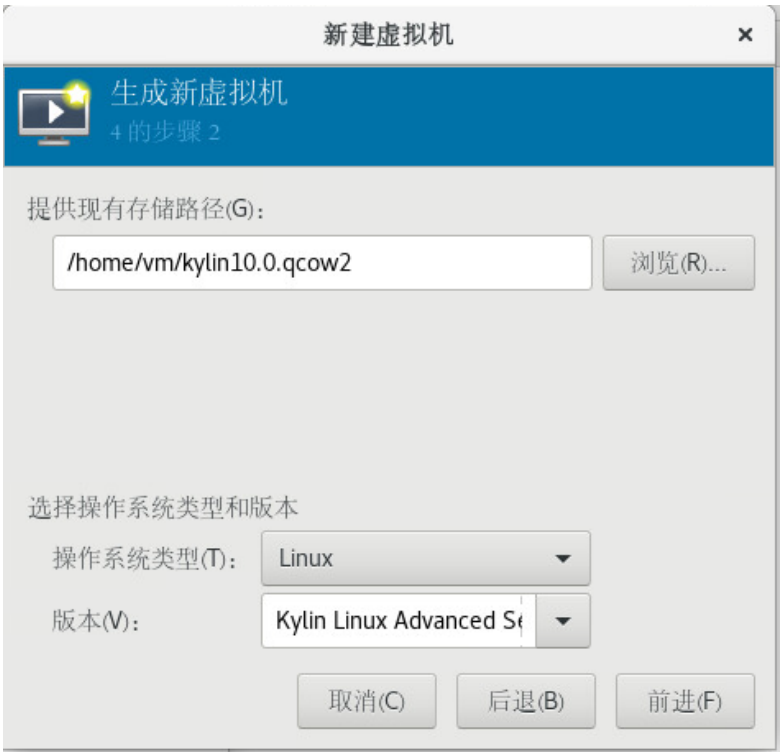


图 5-13 选择本地磁盘镜像

3. 剩余步骤参照章节 5.2.2.2。

5.3. 虚拟机设备配置

在虚拟机“显示虚拟硬件详情”可看到当前虚拟机的配置，并进行修改，部分修改需要在关机后生效。在概况下可以修改虚拟机的名称。点击左侧列表中的设备，在右侧可以看到详情，并且可以删除设备。

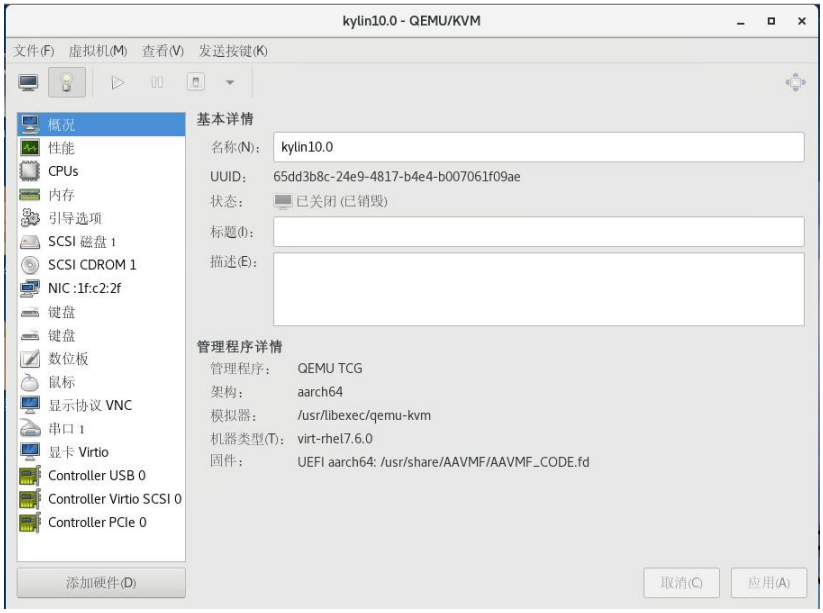


图 5-14 显示虚拟硬件详情

5.3.1. CPU 核数调节

根据本地主机总 CPU 数，以及实际使用情况来调节 CPU 核数。可调节的数范围为：默认分配数量~本地主机最大 CPU 数（实际分配数量根据需求分配）。

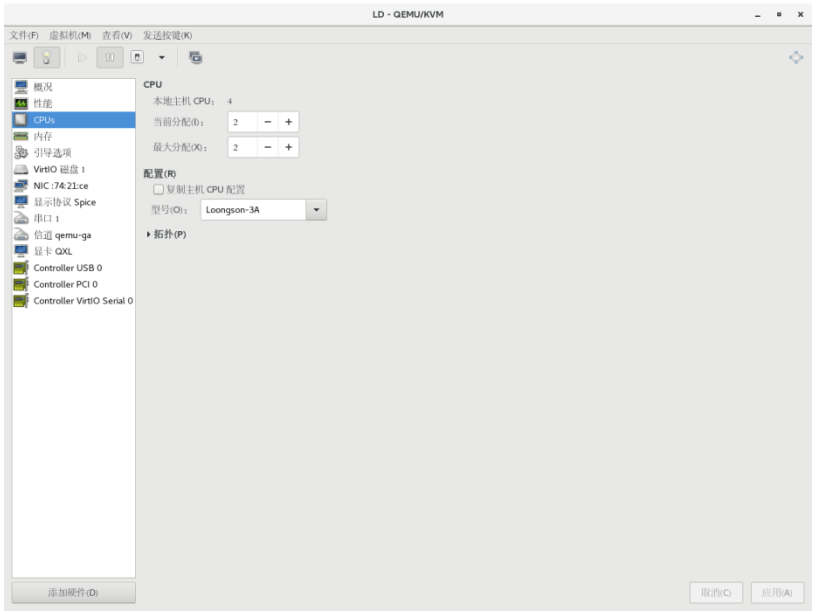


图 5-15 cpu 核数调节

5.3.2. 内存调节

可以根据主机总的内存可用量，以及实际使用情况来调节内存。可调节的范围为：根据实际情况分配内存大小，增减按照 256MB 为一个单位。（**注意：**内存使用还需要根据需要来分配，占用过多会造成本地主机使用卡顿）。

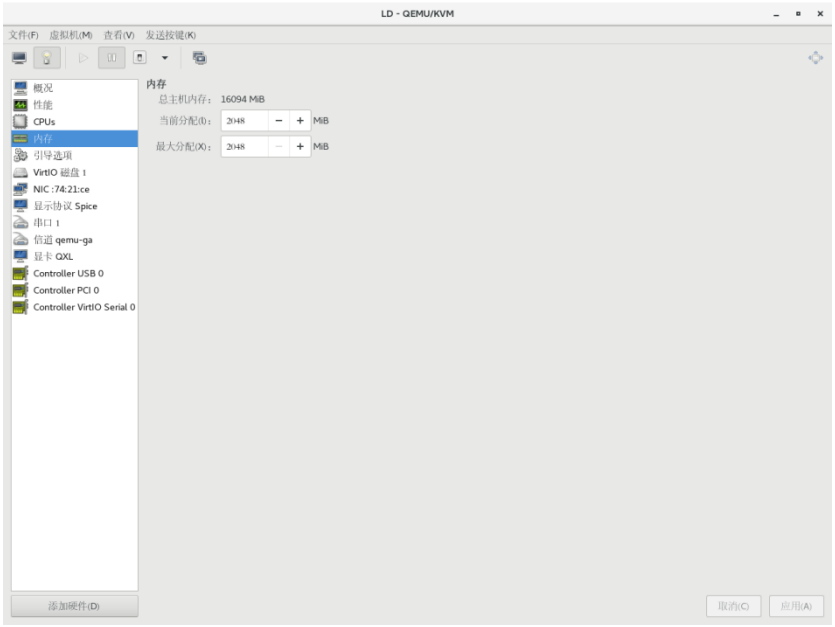


图 5-16 内存调节

5.3.3. 添加存储设备

若后续使用虚拟机过程中，发现磁盘大小不够用，可以在“**添加硬件--存储**”添加虚拟磁盘。

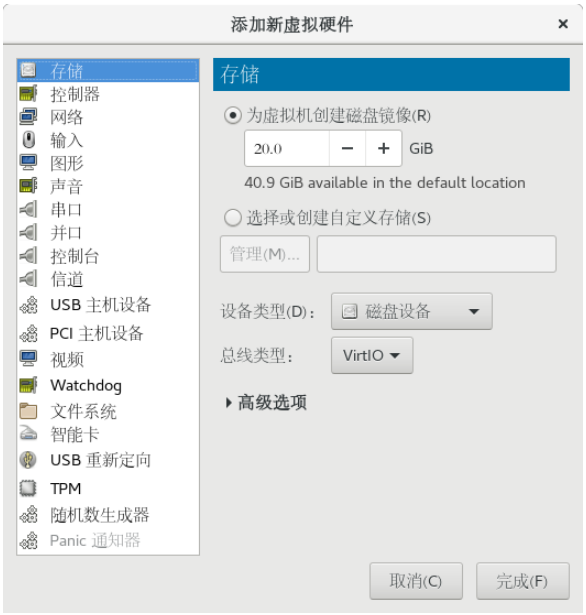


图 5-17 添加存储设备

5.3.4. 添加网络设备

添加硬件中选择网络可以添加网卡设备。

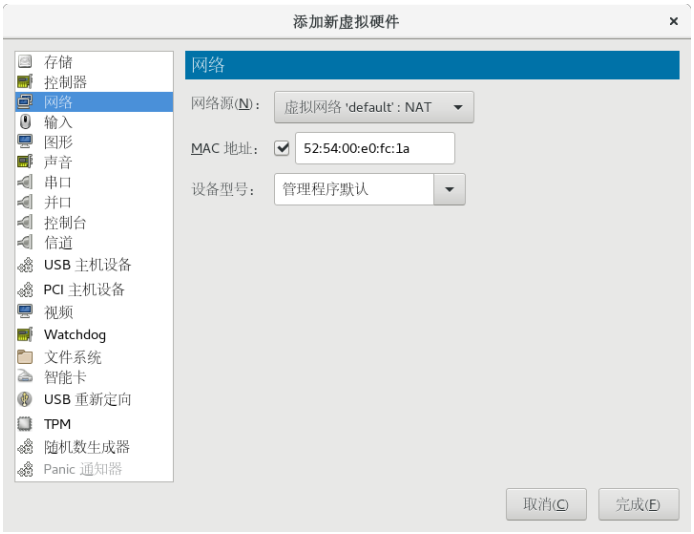


图 5-18 添加网络设备

5.3.5. USB 设备分配

添加硬件中，选择 USB 主机设备，在右侧列表中会列出本地主机的 USB 设备。

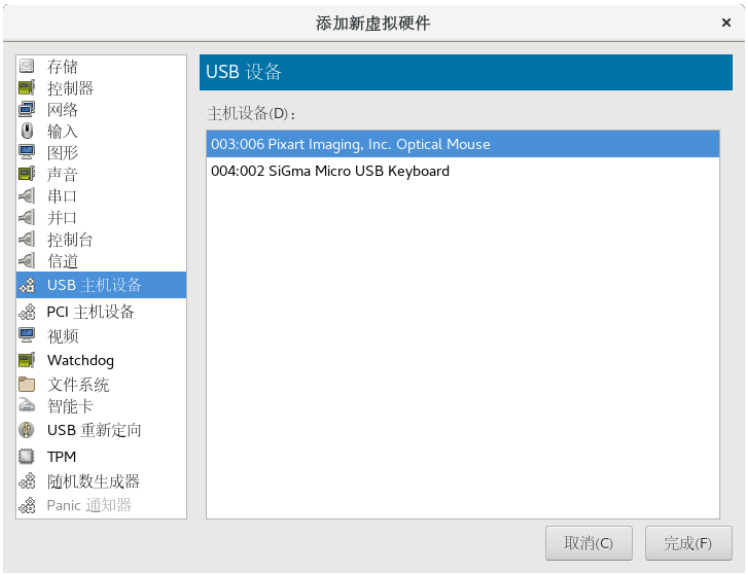


图 5-19 USB 设备分配

选择 USB 重新定向添加 USB 转发器，一个转发器可添加一个客户端 USB 设备，目前只支持”**Spice channel**”类型。

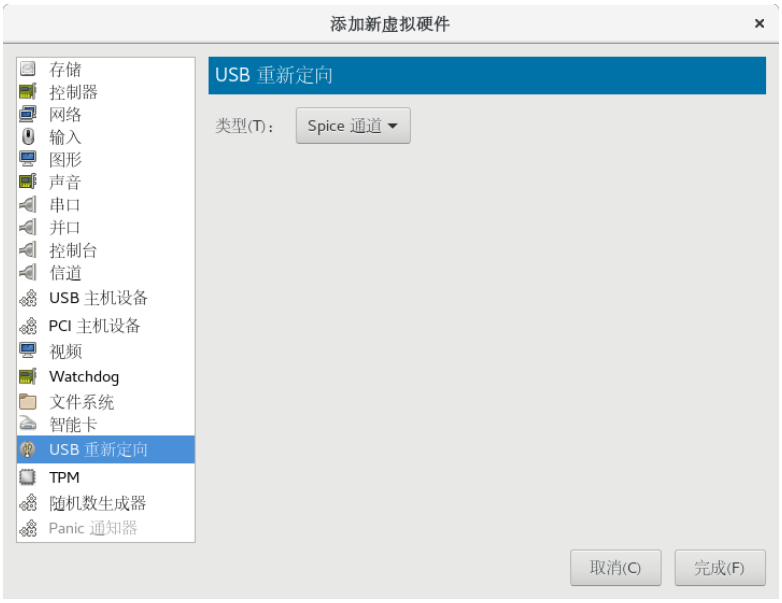


图 5-20 USB 重新定向

5.4. 虚拟机网络设置

Virt-manger 支持两种网络模式：**Nat** 和 **Bridge**。创建虚拟机时，默认选择的是 **Nat** 模式，进入虚拟机后连接网络可以直接使用网络，

以下步骤是 Bridge 的设置步骤：

5.4.1. 设置本地主机网络配置

查看本地主机可以正常使用的网口，例如 enp3s0，配置文件为：/etc/syscon

fig/network-scripts/ifcfg-enp3s0，先复制此文件到相同路径下，命名为 ifcfg-br0。

1. 打开本地主机的网络配置文件

vim /etc/sysconfig/network-scripts/ifcfg-enp3s0

其文件内容为

TYPE=Ethernet

PROXY_METHOD=none

BROWSER_ONLY=no

BOOTPROTO=none

DEFROUTE=yes

IPV4_FAILURE_FATAL=no

IPV6INIT=yes

IPV6_AUTOCONF=yes

IPV6_DEFROUTE=yes

IPV6_FAILURE_FATAL=no

IPV6_ADDR_GEN_MODE=stable-privacy

NAME=enp3s0

UUID=6c3601c7-1e71-49c4-9413-6f7ca9690cc2

DEVICE=enp3s0

ONBOOT=no

IPADDR=10.1.80.145 （本地主机 IP）

PREFIX=24

GATEWAY=10.1.80.254

DNS1=114.114.114.114

2. 配置桥接

以桥接的方式配置网络，修改 ifcfg-enp3s0 文件（有底色的为更改过的内容）。

TYPE=Ethernet

BRIDGE=br0

#PROXY_METHOD=none

#BROWSER_ONLY=no

#BOOTPROTO=none

#DEFROUTE=yes

#IPV4_FAILURE_FATAL=no

#IPV6INIT=yes

#IPV6_AUTOCONF=yes

#IPV6_DEFROUTE=yes

#IPV6_FAILURE_FATAL=no

#IPV6_ADDR_GEN_MODE=stable-privacy

NAME=enp3s0

UUID=6c3601c7-1e71-49c4-9413-6f7ca9690cc2

DEVICE=enp3s0

ONBOOT=yes

#IPADDR=10.1.80.145

#PREFIX=24

#GATEWAY=10.1.80.254

#DNS1=114.114.114.114

修改上一步中拷贝的文件 ifcfg-br0（此文件名可以修改，但注意要和 ifcfg-enp3s0 文件中添加的 BRIDGE=br0 相对应，要修改的配置文件中 NAME 和 DEVICE 也要对应好，有底色的为更改过的内容），ONBOOT 选项可以不做修改，默认是 no，改成 yes 之后，系统在开机时会自动连接网络。

TYPE=Bridge

PROXY_METHOD=none

BROWSER_ONLY=no

BOOTPROTO=static

DEFROUTE=yes

IPV4_FAILURE_FATAL=no

IPV6INIT=yes

IPV6_AUTOCONF=yes

IPV6_DEFROUTE=yes

IPV6_FAILURE_FATAL=no

IPV6_ADDR_GEN_MODE=stable-privacy

NAME=br0

#UUID=6c3601c7-1e71-49c4-9413-6f7ca9690cc2

DEVICE=br0

ONBOOT=yes

IPADDR=10.1.80.145

PREFIX=24

GATEWAY=10.1.80.254

DNS1=114.114.114.114

3. 重启网络服务

systemctl restart network.service

5.4.2. 设置虚拟网络

1. 关闭网络接口

打开 **virt-manager**，在窗口中点击编辑菜单栏中的【连接详情】，选择【网络接口】选项卡，点击窗口左下角红点按钮将 **br0** 接口停止。

图 5-21 关闭网络接口

2. 添加虚拟网络

点击【虚拟网络】选项卡，停止并删除 default 网络，先点击红点按钮停止网络，再点击最右侧按钮删除网络。

图 5-22 删除 default 网络

点击加号添加新建虚拟网络，给新建虚拟网络命名，下一步。

图 5-23 新建虚拟网络命名

设置网段，不勾选【启用 DHCPv4】，下一步。

图 5-24 设置虚拟网络 IP

选择转发模式为【转发到物理网络】

目的：物理设备 br0

模式：路由的

完成（图片中为测试过程截图，网口名称可能会不一致）。

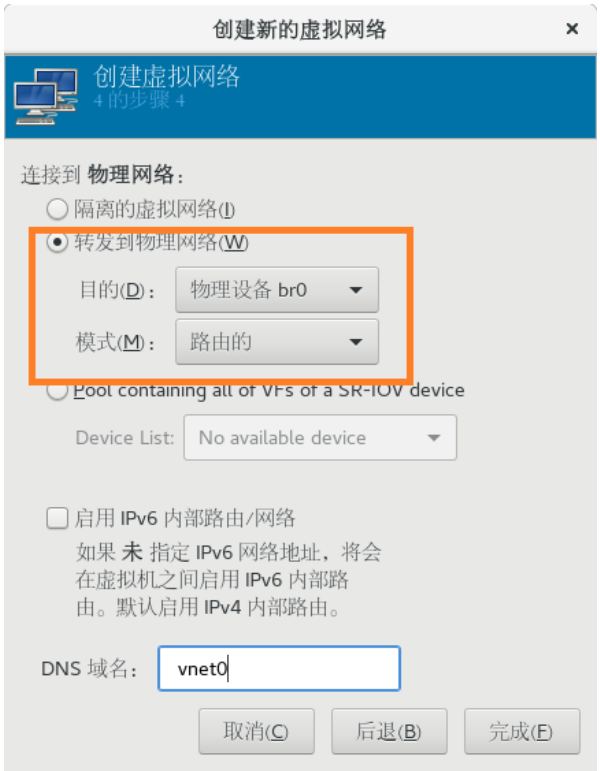


图 5-25 网络模式选择

之后返回【网络接口】选项卡，开启 br0 接口。

5.4.3. 设置虚拟机网络

1. 已安装的虚拟机

在虚拟机详情界面可以更改网络模式，选择【桥接 br0】，应用之后，虚拟机需要重启。

进入系统之后需要手动配置网络，网络需和本地主机在一个网段，重启网络。

若 5.4.1 步骤中将配置文件中的 **ONBOOT** 改为了 **yes**，在虚拟机中手动配置完网络后，需要将网络重新启动一下。

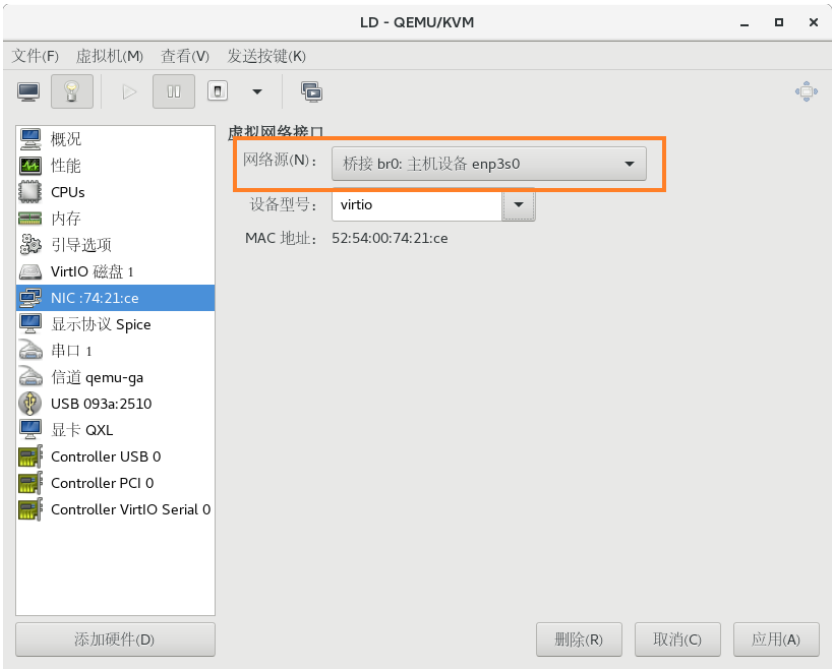


图 5-26 虚拟网络接口设置

2. 新建虚拟机

新建的虚拟机在生成过程中，选择网络为桥接 br0，虚拟机安装完成之后，也需要手动配置网络。

图 5-27 新建虚拟机中的网络选择

5.5. 远程连接虚拟机

VNC 协议

在虚拟机硬件详情页面进行配置，显示协议选择【VNC 服务器】，地址选择【所有接口】。

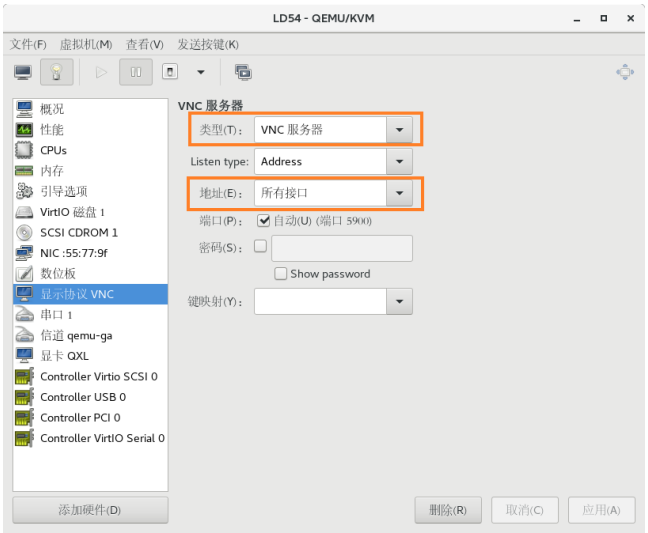


图 5-28 VNC 服务设置

选择使用 VNC 协议进行远程连接时，虚拟机要添加硬件。输入选择 EvTouch **USB** 图形数位板，否则会出现虚拟机里鼠标不对应的问题。

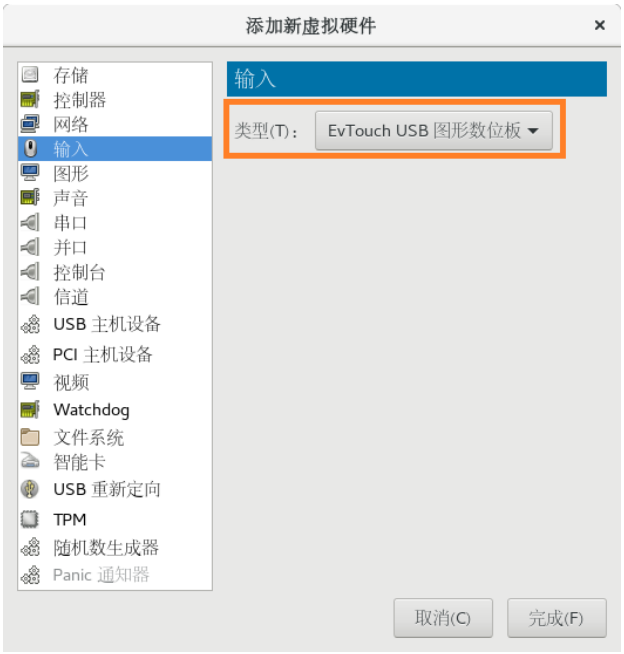


图 5-29 VNC 协议中添加输入硬件

远程通过 remote-viewer 打开客户端；使用 vnc 协议并输入**物理主机**的地址，默认端口号 5900，例如：vnc://10.1.80.145:5900。如果服务器多个虚拟机同时存在，端口号会发生变化。

5.6. 虚拟机迁移

迁移是指把一台物理主机中的一台虚拟机，迁移到另一台物理主机上。

迁移分为热迁移和冷迁移，热迁移是基于共享存储系统的，在虚拟机不中断的情况下进行迁移。而冷迁移只是单纯的把需要迁移的虚拟机的相关配置文件，复制到目的物理主机上。实际应用中，热迁移应用更广泛。

5.6.1. 热迁移

因为热迁移是基于共享存储系统的，所以，需要额外的一台服务器作为存储设备(NFS 服务器)。

5.6.1.1. NFS 服务器

关闭防火墙

systemctl stop firewalld.service

安装/更新 nfs 和 rpcbind 软件

yum -y install nfs-utils rpcbind

创建目录并更改属性

mkdir /home/kvm-fs

chmod 777 /home/kvm-fs

更改/etc/exports 文件

vim /etc/exports

添加以下内容：

/home/kvm-fs *(rw,no_root_squash,no_all_squash,sync)

更新配置

exportfs -r

测试配置是否生效

showmount -e localhost

在/home/kvm-fs 目录下创建虚拟镜像

cd /home/kvm-fs

qemu-img create -f qcow2 test.qcow2 30G

5.6.1.2. 源虚拟机物理主机

1) 关闭防火墙

systemctl stop firewalld.service

2) 创建挂载目录并更改属性

mkdir /home/kvm-fs

chmod 777 /home/kvm-fs

3) 开启 rpcbind 服务

systemctl start rpcbind

4) 挂载到 NFS 服务器上的/home/kvm-fs

mount -t nfs 10.1.xx.xx:/home/kvm-fs /home/kvm-fs

5) 查看是否挂载成功

ls /home/kvm-fs

6) 新建虚拟机，虚拟磁盘路径改为/home/kvm-fs/test.qcow2

7) 建立与目的主机的 ssh 连接

打开虚拟系统管理器，文件菜单栏中的添加链接。选择【连接到远程主机】，方法：SSH，主机名：目的物理主机 IP。

若此步报关于 openssh-askpass 的错误，可能是因为没有安装此软件包，直接 yum 安装即可（KVM 为升级安装的需要把 yum 源改为之前的源）。

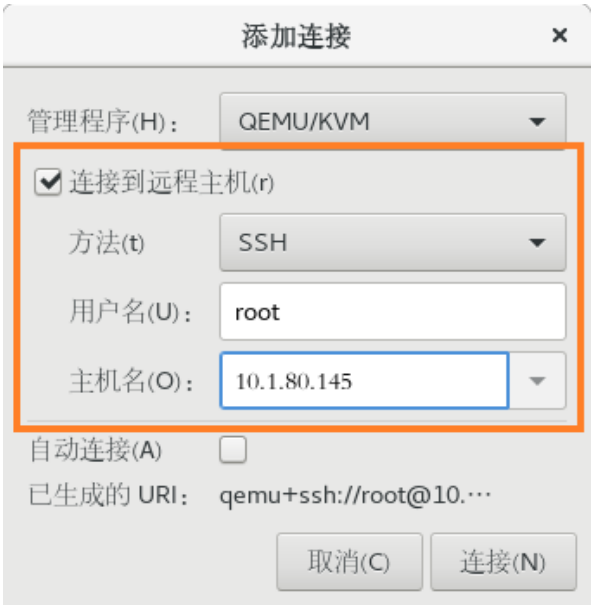


图 5-30 添加连接

5.6.1.3. 目的物理主机

步骤同 5.6.1.2 源虚拟机物理主机配置 1-5 步。

5.6.1.4. 迁移

有两种方式可以打开迁移界面：

1) 虚拟管理界面，右键点击想要迁移的虚拟机，选择迁移

2) 在打开的虚拟机界面，选择菜单栏中虚拟机中的迁移

在弹出的界面填写目的物理机的 IP（目的物理机的主机名不能为 localhost，即目的物理主机需要更改主机名，否则拒绝迁移。）

勾选高级选项中的允许不可靠，否则迁移不成功。之后点击迁移。

图 5-31 迁移

5.6.2. 冷迁移

5.6.2.1. 源虚拟机物理主机

1) 新建虚拟机，虚拟磁盘为本地存储位置

2) 建立与目的物理主机的 ssh 连接

打开虚拟系统管理器，文件菜单栏中的添加链接。选择【连接到远程主机】，方法：SSH，主机名：目的物理主机 IP。

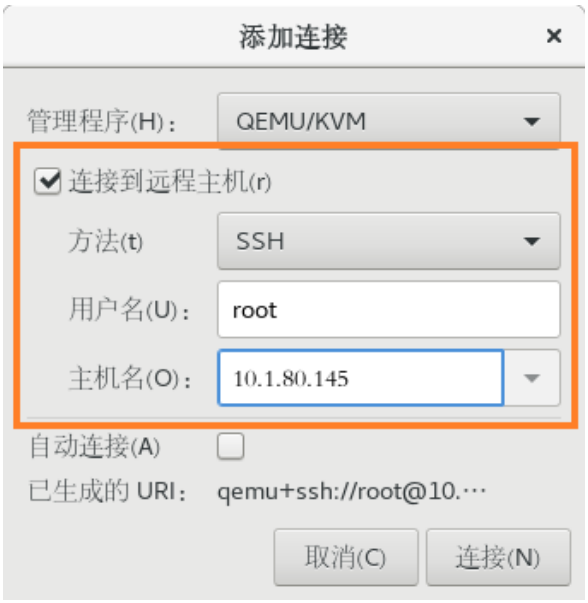


图 5-32 添加连接

5.6.2.2. 目的物理主机

需要与源虚拟机相同路径下创建 qcow2 文件

5.6.2.3. 迁移

方法与热迁移方法相同，有两种打开方式：

1) 虚拟管理界面，右键点击想要迁移的虚拟机，选择迁移

2) 在打开的虚拟机界面，选择菜单栏中虚拟机中的迁移

在弹出的界面填写目的物理机的 IP（目的物理机的主机名不能为 localhost，即目的物理主机需要更改主机名，否则拒绝迁移。）

勾选高级选项中的允许不可靠，否则迁移不成功。之后点击迁移。

图 5-33 虚拟机迁移

第六章 监控和自动化

6.1. 系统监控工具

在配置系统之外，掌握收集基本的系统信息的方法也很重要。譬如，您应该知道如何找出空闲内存的数量、可用硬盘空间，硬盘分区方案，以及正在运行进程的信息等等。本节将说明如何使用几个简单程序来从您的中标麒麟服务器操作系统中检索这类信息。

6.1.1. 查看系统进程

6.1.1.1. 使用 ps 命令

Ps 命令显示系统运行进程的信息。它会生成一个静态列表，列表里的进程是当您执行命令时系统运行的进程的一个快照。如果您想持续更新实时进程列表，您需要使用 top 命令和系统监控应用。

如果想列出当前系统中的所有进程，可以在 shell 里使用如下命令

```
ps ax
```

对于每一个列出的进程，ps ax 命令会显示进程的 PID,进程依附的终端，进程状态，进程占用的 CPU 时间，可执行文件的名字。例如：

```
~]$ ps ax
```

如果想显示进程的所有者，使用如下命令：

```
ps aux
```

除了 ps ax 命令提供的信息外，ps aux 还显示进程拥有者的名字，进程占用 CPU 和内存的百分比，以字节为单位显示虚拟内存大小和未交换物理内存的大小，进程开始运行的时间。

您可以使用 ps 命令和 grep 命令的组合来查看某进程是否在运行。譬如，要判定 Emacs 是否在运行，使用下面这个命令：

```
~]$ ps ax | grep emacs
```

有关可用命令行选项的完整列表，查看 ps(1) man 手册。

6.1.1.2. 使用 top 命令

Top 命令显示系统运行中进程的实时列表。它还会显示附加信息包括系统更新时间，当前 CPU 和内存的使用率，运行的进程总数。这样您可以对进程做更进一步的操作，例如可以给进程排序或者杀死一个进程。

运行 top 命令，在 shell 里输入如下命令：

```
top
```

对于列出的进程，top 命令会显示进程的 PID,进程所有者的名字，进程优先级，进程 NICE 值，进程使用的虚拟内存，进程使用的未交换的物理内存，进程使用的共享内存，进程的状态，进程使用内存和 CPU 的百分比，进程占用 CPU 的时间和可执行文件的名字。

下表可以和 top 一起使用的互动命令：

表格 6-1 交互 top 命令

命令	描述
[Space]	立即刷新显示
[h]	显示帮助屏幕
[k]	杀死某进程。您会被提示输入进程 ID 以及要发送给它的信号
[n]	改变要显示的进程数量。您会被提示输入数量
[u]	按用户排序
[M]	按内存用量排序
[P]	按 CPU 使用率排序
[q]	退出 top

6.1.1.3. 使用系统监控工具

用户可以在系统监控工具的图形界面上查看和查找进程，改变进程的优先级以及杀死进程。

在命令行启动系统监控工具需要在 shell 下输入 gnome-system-monitor。如果使用了 GNOME 桌面，按 Super 键进入活动概述，键入 System Monitor，然后按 Enter 键。出现系统监视器工具。Super 键存在多种伪装，这取决于键盘和其它硬件，但通常作为 Windows 或 command 键，并且通常在空格键的左侧。

点击【进程】标签页查看运行的进程。

进程名	用户	% CPU	ID	内存	磁盘读取总计	磁盘写入总计	磁盘读取	磁盘写入	优先级
abrt-applet	root	0	31066	18.8 MiB	8.0 KiB	0MiB	0KiB/s	0KiB/s	普通
abrt-applet	root	0	13673	18.8 MiB	0MiB	0MiB	0KiB/s	0KiB/s	普通
abrtd	root	0	3047	7.4 MiB	83.7 MiB	268.2 MiB	0KiB/s	0KiB/s	普通
abrt-watch-log	root	0	3050	6.9 MiB	412.0 KiB	0MiB	0KiB/s	0KiB/s	普通
abrt-watch-log	root	0	3051	6.8 MiB	832.0 KiB	0MiB	0KiB/s	0KiB/s	普通
accounts-daemon	root	0	3024	5.1 MiB	2.3 MiB	448.0 KiB	0KiB/s	0KiB/s	普通
acpi_thermal_pm	root	0	108	0MiB	0MiB	0MiB	0KiB/s	0KiB/s	非常高
agetty	root	0	3598	704.0 KiB	92.0 KiB	0MiB	0KiB/s	0KiB/s	普通
atd	root	0	3597	1.8 MiB	1.1 MiB	3.1 MiB	0KiB/s	0KiB/s	普通
at-spi2-registryd	root	0	13397	5.4 MiB	0MiB	0MiB	0KiB/s	0KiB/s	普通
at-spi2-registryd	root	0	30796	5.4 MiB	0MiB	0MiB	0KiB/s	0KiB/s	普通
at-spi-bus-launcher	root	0	13390	3.4 MiB	0MiB	0MiB	0KiB/s	0KiB/s	普通
at-spi-bus-launcher	root	0	30789	3.4 MiB	0MiB	0MiB	0KiB/s	0KiB/s	普通
bash	root	0	11165	2.2 MiB	0MiB	256.0 KiB	0KiB/s	0KiB/s	普通
bash	root	0	11233	2.3 MiB	0MiB	192.0 KiB	0KiB/s	0KiB/s	普通

图 6-1 System Monitor — Processes

对于列出的进程，系统监控工具会显示进程的名字，状态，进程 NICE 值，进程使用内存和 CPU 的百分比，进程的 PID,进程等待的通道和进程会话的其它细节。通过点击进程名字栏可以将进程显示的信息进行升序或则降序排列。

默认系统监控工具会显示当前登录用户的进程列表，通过选择查看菜单下的不同选项，您可以做如下事情：

- 查看活动的进程
- 查看所有进程
- 查看当前登录用户的进程
- 查看进程依赖
- 另外，有两个按钮有如下作用：
- 刷新进程列表
- 选中进程后杀死进程

6.1.2. 查看内存使用情况

6.1.2.1. 使用 free 命令

使用 free 命令可以查看系统空闲和已经使用的内存，在 shell 下输入如下命令:

```
free
```

Free 命令可以提供物理内存和交换空间的信息。它可以显示内存总量，使用的内存大小，空闲内存大小，共享内存大小，buff 的大小和 cache 的大小以及是否可用。例子如下：

```
~]$ free
```

默认 free 以字节显示大小，如果想以兆为单位可以加-m 选项，如下：

```
~]$ free -m
```

6.1.2.2. 使用系统监控工具

系统监控工具的资源标签页可以查看系统中空闲的内存大小和已经使用的大小。

在命令行启动系统监控工具可以在 shell 下输入 gnome-system-momitor。另外，如果使用 GNOME 桌面，按【Super】键进入活动概述，键入【System Monitor】，然后按【Enter】键。出现系统监视器工具。【Super】键存在多种伪装，这取决于键盘和其它硬件，但通常作为 Windows 或 command 键，并且通常在空格键的左侧。

点击资源标签页查看系统内存使用情况。

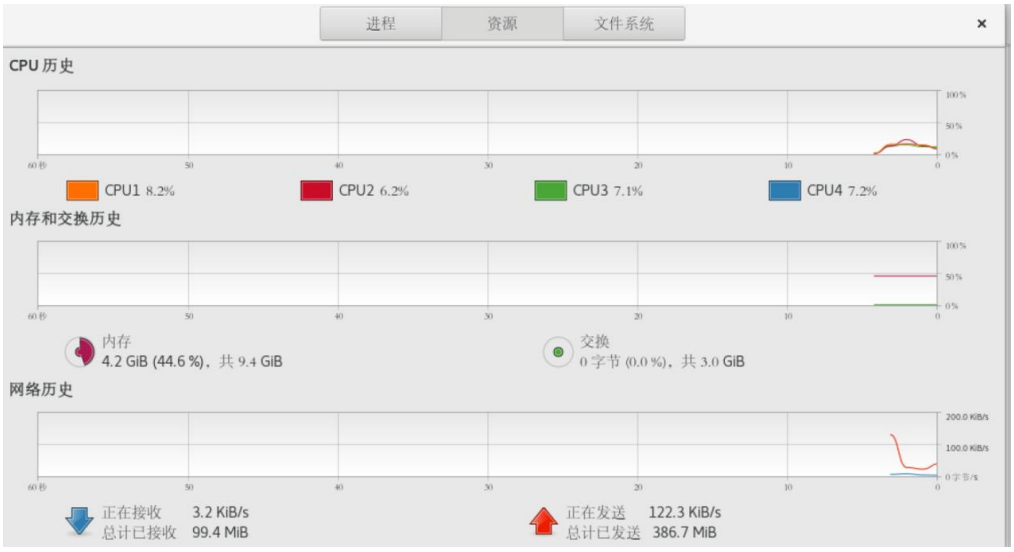


图 6-2 System Monitor — Resources

在内存和交换分区章节，系统监控工具显示内存和交换分区的历史使用图表和物理内存和交换分区的总大小以及使用率。

6.1.3. 查看 CPU 使用

6.1.3.1. 使用系统监控工具

使用系统监控工具的资源标签页可以查看当前系统的 CPU 使用情况。

在命令行启动系统监控工具可以在 shell 下输入 gnome-system-momitor。另外，如果使用 GNOME 桌面，按【Super】键进入活动概述，键入 System Monitor，然后按【Enter】键。出现系统监视器工具。【Super】键存在多种伪装，这取决于键盘和其它硬件，但通常作为 Windows 或 command 键，并且通常在空格键的左侧。

点击资源标签页查看系统的 CPU 使用情况。

在 CPU 章节，系统监控工具显示 CPU 的历史使用图表和 CPU 当前使用率的图表。

6.1.4. 查看块设备和文件系统

6.1.4.1. 使用 lsblk 命令

lsblk 命令可以显示系统可以使用的块设备。它比 blkid 命令提供更多的信息和输出格式控制。它从 udev 读取信息，因此非 root 用户都可以使用。显示块设备列表需要在 shell 输入如下命令：

```
lsblk
```

每一块输出的块设备，lsblk 都显示设备名，主次设备号，设备是否可以删除，设备文件大小，设备是否是只读，设备类型，设备挂载路径。例子如下：

```
~]$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sr0 11:0 1 1024M 0 rom
vda 252:0 0 20G 0 rom
|-vda1 252:1 0 500M 0 part /boot
`-vda2 252:2 0 19.5G 0 part
|-vg_kvm-lv_root (dm-0) 253:0 0 18G 0 lvm /
`-vg_kvm-lv_swap (dm-1) 253:1 0 1.5G 0 lvm [SWAP]
```

默认 lsblk 命令以树形结构输出块设备，如果想获取更多的设备信息添加-l 参数，例如：

```
~]$ lsblk -l
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
```

```
sr0  11:0  1  1024M  0  rom
vda  252:0  0  20G  0  rom
vda1  252:1  0  500M  0  part  /boot
vda2  252:2  0  19.5G  0  part
vg_kvm-lv_root  (dm-0)  253:0  0  18G  0  lvm  /
vg_kvm-lv_swap  (dm-1)  253:1  0  1.5G  0  lvm  [SWAP]
```

有关可用命令行选项的完整列表，查看 lsblk(8) man 手册。

6.1.4.2. 使用 blkid 命令

blkid 命令可以显示可用块设备底层信息。它需要 root 权限，因此非 root 用户只能使用 lsblk 命令，以 root 身份在 shell 输入如下命令：

```
blkid
```

每一个列出的块设备，blkid 会显示它的可用属性，例如 UUID,文件系统类型，卷标签。例子如下：

```
~]# blkid
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4
"
/dev/vda2: UUID="7IvYzk-TnnK-oPjf-ipdD-cofz-DXaJ-gPdgBW"
TYPE="LVM2_member"
/dev/mapper/vg_kvm-lv_root: UUID="a07b967c-71a0-4925-ab02-aebcad2ae
824"
TYPE="ext4"
/dev/mapper/vg_kvm-lv_swap: UUID="d7ef54ca-9c41-4de4-ac1b-4193b0c1
ddb6"
TYPE="swap"
```

默认 blkid 命令会显示所有可用的块设备。如果想显示某一块设备，需要在命令后面加设备名：

```
blkid device_name
```

例如，想显示设备/dev/vda1 的信息，以 root 用户权限输入命令：

```
~]# blkid /dev/vda1
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4
"
```

在上面的命令中加入-p 和-o 选项可以获取更多细节信息，命令同样需要 roo

t 权限。

```
blkid -po udev /dev/vda1
```

例如：

```
~]# bl ki d -po ud ev /d ev/vd a1
ID_FS_UUID=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_UUID_ENC=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_VERSION=1.0
ID_FS_TYPE=ext4
ID_FS_USAGE=filesystem
```

有关可用命令行选项的完整列表，查看 blkid(8) man 手册。

6.1.4.3. 使用 findmnt 命令

findmnt 命令显示系统当前挂载的文件系统，在 shell 里输入如下命令：

```
findmnt
```

每一个列出的文件系统，findmnt 命令显示挂载点，原设备，文件系统类型和有关的挂载选项。

例如：

```
~]$ findmnt
TARGET                          SOURCE             FSTYPE
OPTIONS
/                               /dev/mapper/rhel-root
                                                   xfs
rw,relatime,seclabel,attr2,inode64,noquota
├─/proc                         proc               proc
rw,nosuid,nodev,noexec,relatime
│ ├─/proc/sys/fs/binfmt_misc    systemd-1          autofs
rw,relatime,fd=32,pgrp=1,timeout=300,minproto=5,maxproto=5,direct
│ └─/proc/fs/nfsd               sunrpc             nfsd
rw,relatime
```

```
├─/sys                          sysfs              sysfs
rw,nosuid,nodev,noexec,relatime,seclabel
│ ├─/sys/kernel/security        securityfs         securityfs
rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/cgroup              tmpfs              tmpfs
rw,nosuid,nodev,noexec,seclabel,mode=755
[output truncated]
```

默认 findmnt 以树型结构输出文件系统，如果想获取更多的设备信息添加-l 参数，例如：

```
findmnt -l
```

例如：

```
~]$ findmnt -l
```

您可以选择只输出某种类型的文件系统，使用-t 选项，后面写文件系统类型，例如：

```
findmnt –t type
```

例如，显示所有的 xfs 文件系统:

```
~]$ findmnt -t xfs
```

有关可用命令行选项的完整列表，查看 findmnt(8) man 手册。

6.1.4.4. 使用 df 命令

df 命令输出系统磁盘空间的使用报告，在 shell 里输入如下命令:

```
df
```

每一个列出的文件系统，df 命令都会显示文件系统的名字，大小（以 K 字节为单位），磁盘空间使用率和使用大小，磁盘空间剩余大小和文件挂载点。例子如下：

```
~]$ df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/mapper/vg_kvm-lv_root 18618236 4357360 13315112 25% /
tmpfs 380376 288 380088 1% /dev/shm
/dev/vda1 495844 77029 393215 17% /boot
```

默认 df 命令显示分区大小（以 K 字节为单位），磁盘空间使用总量，磁盘剩余空间可用量（以 K 字节为单位）。想以 M 字节和 G 字节显示需要使用-h 选项，该选项可以以易读方式的格式显示磁盘空间大小。

例如：

```
~]$ df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/vg_kvm-lv_root 18G 4.2G 13G 25% /
tmpfs 372M 288K 372M 1% /dev/shm
/dev/vda1 485M 76M 384M 17% /boot
```

有关可用命令行选项的完整列表，查看 df(1) man 手册。

6.1.4.5. 使用 du 命令

du 命令可以查看文件的大小。du 命令不加参数可以显示每个子目录中文件的大小。例如：

```
~]$ du
14972 ./Downloads
4 ./.mozilla/extensions
4 ./.mozilla/plugins
12 ./.mozilla
15004 .
```

默认情况下，du 命令以千字节为单位显示磁盘的使用情况。如果想以易读方式（MB 和 GB）显示大小，可以添加参数-h。例如;

```
~]$ du -h
15M ./Downloads
4.0K ./.mozilla/extensions
4.0K ./.mozilla/plugins
12K ./.mozilla
15M .
```

在列表末尾 du 命令会显示出当前目录所有文件大小的和，如果只显示目录中文件的总大小可以添加参数-s,例如：

```
~]$ du -sh
15M .
```

有关可用命令行选项的完整列表，查看 du(1) man 手册。

6.1.4.6. 使用系统监控工具

在系统监控工具的文件系统标签页可以以图表的方式查看文件系统和磁盘空间的使用情况。

在命令行启动系统监控工具需要在 shell 下输入 gnome-system-monitor。另外，如果使用 GNOME 桌面，按【Super】键进入活动概述，键入 System Monitor，然后按【Enter】键。出现系统监视器工具。【Super】键存在多种伪装，这取决于键盘和其它硬件，但通常作为 Windows 或 command 键，并且通常在空格键的左侧。

点击文件系统标签页显示文件系统列表，如图：

图 6-3 System Monitor — File Systems

每一个列出的文件系统，系统监控工具都会显示原设备，目标挂载点，文件系统类型和大小，磁盘空间使用大小和未使用大小。

6.1.5. 查看硬件信息

6.1.5.1. 使用 lspci 命令

lspci 命令可以显示 PCI 总线上的设备信息，显示所有的 PCI 设备在 shell 里输入如下命令：

```
~]$ lspci
```

您可以使用-v 选项输出设备详细信息，使用-vv 选项输出设备更详细的信息。

```
~]$ lspci -v
```

有关可用命令行选项的完整列表，查看 lspci(8) man 手册。

6.1.5.2. 使用 lsusb 命令

lsusb 命令可以显示 usb 设备信息。显示所有的 usb 设备信息在 shell 里输入如下命令：

```
~]$ lsusb
```

您可以添加-v 参数显示设备的详细信息。

```
~]$ lsusb -v
```

有关可用命令行选项的完整列表，查看 lsusb(8) man 手册。

6.1.5.3. 使用 lscpu 命令

lscpu 命令显示当前系统的 cpu 信息，信息包括 cpu 数目，架构，字节序等信息，在 shell 输入命令：

```
~]$ lscpu
```

有关可用命令行选项的完整列表，查看 lscpu(1) man 手册。

6.1.6. 检查硬件错误

中标麒麟高级服务器操作系统软件引入了新的硬件事件报告机制。这个机制会为 DIMMs 收集内存错误和错误检查和纠错机制报告的错误，并且把错误向用户空间报告。用户空间的守护进程 rasdaemon 会捕获和处理这些由内核机制跟踪的有可靠性可用性可维护性的错误事件，并记录它们。这个功能以前由 edac-utils 提供，现在由守护进程 rasdaemon 提供。

安装 rasdaemon 需要 root 权限。

```
~]# yum install rasdaemon
```

启动服务命令如下

```
~]# systemctl start rasdaemon
```

输入下面的命令查看命令的各种选项

```
~]$ rasdaemon --help
```

命令在 rasdaemon(8) man 手册里页也有描述。

ras-mc-ctl 工具提供了一种可以使用 EDAC 驱动的方法，输入如下命令可以查看命令的选项。

```
~]$ ras-mc-ctl --help
```

命令在 ras-mc-ctl (8) man 手册里页也有描述。

6.1.7. 使用 Net-SNMP 监控性能

中标麒麟高级服务器操作系统软件包括 Net-SNMP 软件套件，其中包括一个灵活的可扩展的简单网络管理协议（SNMP）代理。该代理及其关联的实用程序可以被用于从大量的系统中提供性能数据给一系列工具，这些工具都支持 SNMP 协议。

本节提供了关于配置 Net-SNMP 代理通过网络安全地提供性能数据，使用 SNMP 协议检索数据和扩展 SNMP 代理以提供自定义的性能指标。

6.1.7.1. 安装 Net-SNMP

Net-SNMP 软件套件可作为中标麒麟高级服务器操作系统软件 V7.0 发行版的一组 RPM 软件包。表格 6-2 可用 Net-SNMP 软件包概述了每个包和它们的内容。

表格 6-2 可用 Net-SNMP 软件包

软件包	提供商

net-snmp	SNMP 代理守护进程和文档。输出性能数据需要这个包。
net-snmp-libs	netsnmp 库和捆绑的管理信息库（MIBs）。输出性能数据需要这个包。
net-snmp-utils	SNMP 客户端例如 snmpget 和 snmpwalk。 需要该软件包以查询 SNMP 系统的性能数据。
net-snmp-perl	mib2c 程序和 NetSNMP Perl 模块。此包是由可选通道提供的。
net-snmp-python	Python 的 SNMP 客户端库。此包是由可选通道提供的。

安装任何一个软件包按如下方式使用 yum 命令：

```
yum install package…
```

例如，安装在本节的其余部分中使用的 SNMP 代理守护进程和 SNMP 客户端，以 root 身份在 shell 键入如下命令：

```
~]# yum install net-snmp net-snmp-libs net-snmp-utils
```

6.1.7.2. 运行 Net-SNMP 守护进程

net-snmp 软件包中包含 SNMP 代理守护进程 snmpd。本节提供有关如何启动，停止和重新启动 snmpd 服务的信息。有关在中标麒麟高级服务器操作系统软件 V7.0 系统的管理服务的详细信息请参阅 3.1 使用 systemd 管理系统服务。

启动服务

运行 snmpd 服务以 root 身份在 shell 键入如下命令：

```
systemctl start snmpd.service
```

配置服务使其在系统启动后自动开始运行使用如下命令：

```
systemctl enable snmpd.service
```

停止服务

停止 snmpd 服务以 root 身份在 shell 键入如下命令：

```
systemctl stop snmpd.service
```

配置服务使其在系统启动后并不开始运行，使用如下命令：

```
systemctl disable snmpd.service
```

重启服务

重启 snmpd 服务以 root 身份在 shell 键入如下命令：

```
systemctl restart snmpd.service
```

该命令停止服务并快速重启服务。要仅重新加载配置，而无需停止服务，运行以下命令：

```
systemctl reload snmpd.service
```

这会运行 snmp 服务去重新加载配置。

6.1.7.3. 配置 Net-SNMP

要修改 Net-SNMP 代理守护进程的配置，编辑/etc/snmp/snmpd.conf 配置文件。中标麒麟高级服务器操作系统软件 V7.0 包含的默认的 snmpd.conf 文件被大量注释，并可以作为一个很好的代理配置文件的起点。

本节主要介绍两种常见的任务：设置系统信息并配置认证。有关可用的配置指令的详细信息，请参阅 snmpd.conf(5)手册页。此外，在 net-snmp 软件包里有一个名为 snmpd.conf 的实用程序，可以用来以交互方式生成代理配置。

需要注意的是 net-snmp-util 软件包必须被安装才能使用本节所述的 snmpwalk 实用程序。

```
systemctl reload snmpd.service
```

设置系统信息

Net-SNMP 通过系统树提供了一些基本的系统信息。例如，下面的 snmpwalk 的命令显示具有默认代理配置系统树。

```
~]# snmpwalk -v2c -c public localhost system
```

缺省情况下，sysName 对象被设置为主机名。sysLocation 和 sysContact 对象可以在/etc/snmp/snmpd.conf 文件通过改变 syslocation 和 syscontact 的值被配置。例如:

```
syslocation Datacenter, Row 4, Rack 3
syscontact UNIX Admin <admin@ example.com>
```

更改配置文件后，重新加载配置，并通过再次运行 snmpwalk 命令测试一下配置是否生效。

```
~]# systemctl reload snmp.service
~]# snmpwalk -v2c -c public localhost system
```

配置权限

Net-SNMP 代理守护程序支持所有三个版本的 SNMP 协议。前两个版本（1 和 2c）通过使用字符串提供简单的身份验证。该字符串是代理人和任何客户端实用程序之间共享的秘密。该字符串在网络上用明文传递，这认为是不安全的。SNMP 协议第 3 版支持使用多种协议的用户认证和信息加密。Net-SNMP 代理还支持 SSH 通道，使用 X.509 证书的 TLS 认证和 Kerberos 认证。

配置 SNMP 版本 2c

要配置 SNMP 版本 2c 的社区，在/etc/snmp/snmpd.conf 配置文件下使用 RO 社区或 RW 社区指令。指令的格式是如下：

```
directive community [source [OID]]
```

其中，community 要使用社区字符串，source 是 IP 地址或子网， OID 对 SNMP 树提供访问属性。例如，下面的指令对客户端提供只读的系统树访问，该客户端使用本地计算机上的社区字符串“kylin”：

```
rocommunity kylin 127.0.0.1 .1.3.6.1.2.1.1
```

使用 snmpwalk 命令并添加-v 和-c 参数可以测试配置。

```
~]# snmpwalk -v2c -c kylin localhost system
```

配置 SNMP 版本 3

要配置 SNMP 版本 3，使用 net-snmp-create-v3-user 命令。此命令添加条目到/var/lib/net-snmp/snmpd.conf 和/etc/snmp/snmpd.conf 文件，这两个文件可以创建用户并授权访问。注意，net-snmp-create-v3-user 命令只有代理没有运行时才能运行。下面的示例创建了“admin”用户，密码“kylinsnmp”：

```
~]# systemctl stop snmpd.service
~]# net-snmp-create-v3-user
```

net-snmp-create-v3-user 命令添加 rwuser 指令（或当提供的-ro 命令行选项使用 rouser）到/etc/snmp/snmpd.conf 中和添加 rwcommunity 和 rocommunity 的格式相同：

```
directive user [no auth|auth|priv] [OID]
```

其中，user 是用户名，OID 是 SNMP 树提供访问。默认情况下，Net-SNMP 代理仅允许已通过授权的请求（auth 选项）。noauth 选项可以允许未经授权的请求，priv 选项强制使用加密,authpriv 选项指定的请求必须通过授权和回复被加密。

例如，下面一行表示授予用户 “admin”读写访问整个树：

```
rwuser admin authpriv .1
```

为了测试配置，在您的用户目录下创建一个 snmp 目录，在该目录下创建 snmp.conf 文件（~ /.snmp/snmp.conf），使用如下命令：

```
defVersion 3
defSecurityLevel authPriv
defSecurityName admin
defPassphrase kylinsnmp
```

snmpwalk 命令在查询代理时会使用这些授权设置。

```
~]$ snmpwalk -v3 localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 3.10.0-
123.el7.x86_64 #1 SMP Mon May 5 11:16:57 EDT 2014 x86_64
[output truncated]
```

6.1.7.4. 检索数据性能

中标麒麟高级服务器操作系统软件的 Net-SNMP 代理在 SNMP 协议上提供了多种性能信息。此外，可以通过代理查询系统中安装 RPM 包的列表，系统当前运行的进程列表和系统的网络配置。

本节提供了关于 OIDs 在 SNMP 上的性能概述。它假定系统已安装 net-snmp-utils 软件包，并且用户被授予访问 SNMP 树，如 6.1.7.3 配置权限中描述的。

硬件配置

包括 Net-SNMP 的 Host Resources MIB 提供了主机客户端程序的硬件和软件配置。表格 6-3 可用的 OIDs 总结了在 MIB 下可以获取的不同的 OID 。

表格 6-3 可用的 OIDs

OID	描述
HOST-RESOURCES-MIB::hrSystem	包含一般系统信息，如运行时间，用户的数目，和运行的进程的数目。
HOST-RESOURCES-MIB::hrStorage	包含内存和文件系统使用情况数据。
HOST-RESOURCES-MIB::hrDevices	包含所有的处理器，网络设备和文件

	系统的列表。
HOST-RESOURCES-MIB::hrSWRun	包含所有正在运行的进程的列表。
HOST-RESOURCES-MIB:: hrSWRunPerf	包含进程表中的内存和 CPU 统计信息，进程表来自于 HOST-RESOURCES-MIB::hrSWRun。
HOST-RESOURCES-MIB::hrSWInstalled	包含 RPM 数据库的列表。

在 Host Resources MIB 中，还有一些可用于检索的可用信息的 SNMP 表。下面的例子列出了 HOST-RESOURCES-MIB:: hrFSTable:

```
~]$ snmptable -Cb localhost HOST-RESOURCES-MIB::hrFSTable
SNMP table: HOST-RESOURCES-MIB::hrFSTable

 Index MountPoint RemoteMountPoint                                  Type
    Access Bootable StorageIndex LastFullBackupDate LastPartialBackupDate
     1        "/"                "" HOST-RESOURCES-TYPES::hrFSLinuxExt2
 readWrite     true           31      0-1-1,0:0:0.0         0-1-1,0:0:0.0
     5 "/dev/shm"                ""     HOST-RESOURCES-TYPES::hrFSOther
 readWrite    false           35      0-1-1,0:0:0.0         0-1-1,0:0:0.0
     6    "/boot"                "" HOST-RESOURCES-TYPES::hrFSLinuxExt2
 readWrite    false           36      0-1-1,0:0:0.0         0-1-1,0:0:0.0
```

关于 HOST -RESOURCES-MIB 的更多信息，可以查看/usr/share/snmp/mi bs/HOST-RESOURCES-MIB. txt 文件。

CPU 和内存信息

UCD-SNMP-MIB 的大多数系统性能数据是可用的。systemStats OID 提供了一些处理器的使用计数器：

```
~]$ snmpwalk localhost UCD -SNMP -MIB: : systemStats
```

特别是， ssCpuRawUser, ssCpuRawSystem, ssCpuRawWait 和 ssCpuRawIdle OIDs 提供的计数器在确定系统是否要花费了大量的处理器时间在内核空间或用户空间或 I/O 时非常有用。ssRawSwapIn 和 ssRawSwapOut 可确定系统内存是否耗尽时非常有用。

UCD-SNMP-MIB:: memory OID 下很多可用的内存信息和 free 命令类似：

```
~]$ snmpwalk localhost UCD -SNMP -MIB: : memory
```

UCD-SNMP-MIB 的负载均衡是可用的。SNMP 的 UCD-SNMP-MIB:: laTable 会列出 1 分钟，5 分钟和 15 分钟的负载均衡值。

```
~]$ snmptable localhost UCD-SNMP-MIB::laTable
SNMP table: UCD-SNMP-MIB::laTable

 laIndex laNames laLoad laConfig laLoadInt laLoadFloat laErrorFlag
laErrMessage
       1  Load-1   0.00    12.00         0    0.000000     noError
       2  Load-5   0.00    12.00         0    0.000000     noError
       3 Load-15   0.00    12.00         0    0.000000     noError
```

文件系统和磁盘信息

Host Resources MIB 提供文件系统的大小和使用信息。每个文件系统（以及各内存池）在 HO ST-RESOURCES-MIB: : hrStorageTable 表中有一个入口：

```
~]$ snmptable -Cb localhost HOST-RESOURCES-MIB::hrStorageTable
SNMP table: HOST-RESOURCES-MIB::hrStorageTable

 Index                                          Type           Descr
AllocationUnits    Size   Used AllocationFailures
     1           HOST-RESOURCES-TYPES::hrStorageRam Physical memory
1024 Bytes 1021588 388064                  ?
     3 HOST-RESOURCES-TYPES::hrStorageVirtualMemory  Virtual memory
1024 Bytes 2045580 388064                  ?
     6          HOST-RESOURCES-TYPES::hrStorageOther  Memory buffers
1024 Bytes 1021588  31048                  ?
     7          HOST-RESOURCES-TYPES::hrStorageOther   Cached memory
1024 Bytes  216604 216604                  ?
    10 HOST-RESOURCES-TYPES::hrStorageVirtualMemory      Swap space
1024 Bytes 1023992      0                  ?
    31     HOST-RESOURCES-TYPES::hrStorageFixedDisk              /
4096 Bytes 2277614 250391                  ?
    35     HOST-RESOURCES-TYPES::hrStorageFixedDisk        /dev/shm
4096 Bytes  127698      0                  ?
    36     HOST-RESOURCES-TYPES::hrStorageFixedDisk           /boot
1024 Bytes  198337  26694                  ?
```

HOST-RESOURCES-MIB: : hrStorageSize 和 HOST-RESOURCESMIB:: hr StorageUsed 中的 ODI 被用来计算挂载文件系统的剩余容量。

I/O 数据在 UCD-SNMP-MIB::systemStats 和 UCD-DISKIO-MIB::diskIOT abl e 表中都是可用的。后者提供了更详细的数据，后者的 OID 有 diskIONReadX 和 diskIO NWrittenX 其提供计数器，用于在系统启动时记录读取和写入块设备的字节数。

```
~]$ snmptable -Cb localhost UCD-DISKIO-MIB::diskIOTable
SNMP table: UCD-DISKIO-MIB::diskIOTable

 Index Device      NRead  NWritten Reads Writes LA1 LA5 LA15     NReadX
NWrittenX
...
    25    sda 216886272 139109376 16409   4894   ?   ?    ? 216886272
139109376
    26   sda1   2455552      5120   613      2   ?   ?    ?   2455552
5120
    27   sda2   1486848         0   332      0   ?   ?    ?   1486848
0
    28   sda3 212321280 139104256 15312   4871   ?   ?    ? 212321280
139104256
```

网络信息

Interfaces MIB 提供网络设备信息。IF-MIB::ifTable 提供了一个 SNMP 表，

系统上每个接口及其配置和各种数据包计数器在这个表里都一个条目。下面的例子显示 ifTabel 的部分列，它显示了系统上的两个物理网络接口：

```
~]$ snmptable -Cb localhost IF-MIB: : ifTable
```

网络流量是根据 IF-MIB: : ifOutOctets 和 IF-MIB: : ifInOctets 提供。下面 SNMP 查询将检索此系统上的每一个接口的网络流量。

```
~]$ snmpwalk localhost IF-MIB: : ifDescr
```

扩展的 Net-SNMP

Net-SNMP 代理可以扩展到提供除原生系统度量的应用度量。这使得容量规划和性能问题的故障排除。例如，在测试中知道邮件系统每 15 分钟有 5 分钟处在负载均衡状态是有帮助的，但更有用的是知道邮件系统在每秒处理 8000 个消息时它 15 分钟负载均衡是多少。当应用程序负载的度量可通过相同的接口作为系统的度量，这也允许在不同的负载影响的情况下对系统性能可视化（例如，每增加 10,000 条信息会增加平均负载线至 100,000）。

包括中标麒麟高级服务器操作系统软件 V7.0 的一些应用程序通过 SNMP 扩展 Net-SNMP 代理以提供应用度量。有几种方法来扩展代理定制应用程序。本节介绍通过 shell 脚本和可选的 Perl 插件来扩展代理。假设系统中 net-snmp-utils 和 net-snmp-perl 包已经安装，同时用户被授权访问 SNMP 树。

使用 shell 脚本扩展 Net-SNMP

在 Net-SNMP 代理提供了一个扩展 MIB（NET-SNMP-EXTEND-MIB），可以用来查询任意的 shell 脚本。要指定的 shell 脚本来运行，可以在/etc/snmp/snmpd.conf 文件使用 extend 指令。一旦定义，代理将提供退出码和 在 SNMP 上命令的任何输出。下面的例子演示了这种机制，该脚本确定在进程表中 httpd 进程的数量。

注意:

Net-SNMP 代理提供了一个内建的机制通过 proc 命令检查进程表。更多的信息可以查看 snmpd.conf (5)手册页。

下面 shell 脚本的退出码是在给定时间点的系统上运行的 httpd 进程数。

```
#!/bin/sh
NUMPIDS=`pgrep httpd | wc -l`
exit $NUMPIDS
```

为了使这个脚本在 SNMP 上可用，将脚本复制到系统路径上的某个位置，设置可执行位，并添加 extend 指令到/ etc/snmp/snmp.cof 文件。添加 extend 指令

的格式如下：

```
extend name prog args
```

name 是标识 extend 的字符串，prog 是要运行的程序，args 是传入程序的参数。例如，如果上面的 shell 脚本复制到/usr/local/bin/check_apache.sh，下面的指令会添加脚本到 SNMP 树。

```
extend httpd_pids /bin/sh /usr/local/bin/check_apache.sh
```

脚本可以在 NET -SNMP -EXT END -MIB: : nsExtend Objects 中查询到。

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING:
/usr/local/bin/check_apache.sh
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."httpd_pids" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendExecType."httpd_pids" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."httpd_pids" = INTEGER: run-on-
read(1)
NET-SNMP-EXTEND-MIB::nsExtendStorage."httpd_pids" = INTEGER:
permanent(4)
NET-SNMP-EXTEND-MIB::nsExtendStatus."httpd_pids" = INTEGER: active(1)
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutNumLines."httpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING:
```

请注意，退出码设置为整型（示例中退出码为 8），其它任何输出为字符串类型。为了显示整数的多重度量，extend 指令提供了不同的参数。例如，下面的脚本可以被用于确定匹配任意字符串的进程数，并且也将输出一个文本字符串来表示进程数:

```
#!/bin/sh
PATTERN=$1
NUMPIDS=`pgrep $PATTERN | wc -l`
echo "There are $NUMPIDS $PATTERN processes."
exit $NUMPIDS
```

当上面的脚本被拷贝到/usr/local/bin/check_proc.sh 下，执行下面/etc/snmp/snmpd.conf 文件中的指令会显示出 httpd 和 snmpd 的进程数。

```
extend httpd_pids /bin/sh /usr/local/bin/check_proc.sh httpd
extend snmpd_pids /bin/sh /usr/local/bin/check_proc.sh snmpd
```

下面的例子显示 snmpwalk 命令对 nsExtend Objects OID 的输出：

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 2
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendCommand."snmpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING:
/usr/local/bin/check_proc.sh httpd
NET-SNMP-EXTEND-MIB::nsExtendArgs."snmpd_pids" = STRING:
/usr/local/bin/check_proc.sh snmpd
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."snmpd_pids" = STRING:
...
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendResult."snmpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING: There are 8
httpd processes.
NET-SNMP-EXTEND-MIB::nsExtendOutLine."snmpd_pids".1 = STRING: There are 1
snmpd processes.
```

警告：

整数退出码被限制在 0-255 的范围。对于超过 256 的值，既可以使用脚本的标准输出（被认为是字符串）或扩展代理的不同方法。

6.2. OpenLMI

开放式 linux 管理设备，通常被简称为 OpenLMI，它是管理 linux 系统的通用基础设备。它构建于现有工具基础之上，并为了隐藏来自系统管理员的底层系统大量复杂性而作为抽象层运转。OpenLMI 和一系列可本地或远程登陆的服务共同发行，提供多语言绑定、标准 APIs 和可用于管理监测硬件、操作系统、系统服务的标准脚本。

6.2.1. 关于 OpenLMI

OpenLMI 旨在对在物理机和虚拟机上运行中标麒麟高级服务器操作系统软件 V7.0 系统的产品服务器提供一个通用管理界面，它由以下三个部分组成：

1) 系统管理代理——这些代理都安装了管理系统，实现一个呈现给标准对象中间商的对象模型。在 OpenLMI 实现的最初的代理包括存储配置和网络配置，但后来的工作会处理系统管理的附加元素。系统管理代理通常被称为通用信息模型提供商或者 CIM 提供者。

2) 一个标准对象中间商——对象中间商管理代理并给他们提供接口。标准对象中间商也被称为 CIM 对象监控或 CIMOM。

3) 客户端应用程序和脚本——客户端应用程序和脚本通过标准对象中间商调用系统管理代理。

OpenLMI 项目通过提供一个底层接口实施现有管理措施，这些底层接口可以被脚本或系统管理控制台使用。和 OpenLMI 一起发行的接口包括 C，C ++，Python，Java 和一个交互式命令行客户端，他们都提供相同的完全访问每个代理实现的功能。这可以确保您始终能够访问完全相同的功能，无论您决定使用哪个编程接口。

6.2.1.1. 主要特性

下面是在您的系统上安装和使用 OpenLMI 的主要优势：

- OpenLMI 提供一个标准接口来配置，管理和监控本地和远程系统。
- 它允许您配置，管理和监控在物理机和虚拟机上运行的产品服务器。
- 和它共同发行的一系列 CIM 提供商可以使您配置，管理和监控存储设备和进行复杂的网络工作。
- 它可以让您从 C，C++，Python 和 Java 程序调用系统管理功能，包括 LMIShell，这提供了一个命令行界面。
- 它是一个基于开放的行业标准的免费软件。

6.2.1.2. 管理能力

OpenLMI 的主要功能包括存储设备，网络，系统服务的管理，用户帐户，硬件和软件配置，电源管理和交互使用动态目录。随中标麒麟高级服务器操作系统软件 V7.0 绑定发行的 CIM 提供商的完整列表，请参阅表格 6-4 可用 CIM 提供商。

表格 6-4 可用 CIM 提供商

包名	描述
openlmi-account	用于管理用户帐户的 CIM 提供商。
openlmi-logicalfile	用于读取文件和目录的 CIM 提供商
openlmi-networking	用于网络管理的 CIM 提供商
openlmi-powermanagement	用于电源管理的 CIM 提供商
openlmi-service	用于管理系统服务的 CIM 提供商
openlmi-storage	用于存储管理的 CIM 提供商
openlmi-fan	用于控制 cpu 风扇的 CIM 提供商
openlmi-hardware	用于接收硬件信息的 CIM 提供商
openlmi-realmd	用于配置 realmd 的 CIM 提供商
openlmi-software [a]	用于软件管理的 CIM 提供商
[a] 在中标麒麟高级服务器操作系统软件中，OpenLMI 软件供应商是作为一个技术预览。此提供商是全功能的，但有一个性能问题，当列出了大量的软件包时，可能消耗的内存和时间过多。要解决这个问题，搜索返回尽可能少的包。	

6.2.2. 安装 OpenLMI

OpenLMI 与一组 RPM 包绑定发行，其中包括 CIMOM，独立 CIM 提供商，以及客户端应用程序。这使您可以区分管理端和客户端系统，只安装您需要的组

件。

6.2.2.1. 在管理端安装 OpenLMI

管理端是您通过使用 OpenLMI 客户端工具实施监控，管理的。

要在管理端上安装 OpenLMI，请完成以下步骤：

1) 安装 tog-pegasus 包，需要以 root 身份在 shell 里输入如下命令：

```
yum install tog-pegasus
```

此命令安装 OpenPegasus CIMOM 和它所有的依赖组件，并为 pegasus 使用者创建用户帐户。

2) 通过 root 身份运行以下命令，安装所需 CIM 提供商：

```
yum install openlmi-
{storag e,networking ,service,account,powermanag ement}
```

此命令安装 CIM 提供商，可用于存储，网络，服务，帐户和电源管理。随中标麒麟高级服务器操作系统软件 V7.0 绑定发行的 CIM 提供商的完整列表，请参阅表格 6-4 可用 CIM 提供商表格。

3) 编辑 the /etc/Pegasus/access.conf 自定义许连接到的 OpenPegasus CIMOM 用户的列表。默认情况下，只允许 peg asus 用户可以本地和远程访问 CIMOM。要激活此用户帐户以 root 身份运行下面命令，并设置密码：

```
passwd pegasus
```

4) 通过激活 to g -peg asus. servi ce 来启动 OpenPegasus CIMOM，以 root 身份运行如下命令：

```
systemctl start tog-pegasus.service
```

配置 to g -peg asus. servi ce 服务开机自启动：

```
systemctl enable tog-pegasus.service
```

5) 如果您想从远程与管理系统进行交互，在端口 5989 使能 TCP 链接(wbem-https)，以 ROOT 身份运行如下命令：

```
firewall -cmd --add -port 5989 /tcp
```

为 tcp 链接永久的打开 5989 端口：

```
firewall -cmd --permanent --add -port 5989 /tcp
```

现在，您可以通过使用 OpenLMI 客户端工具连接到管理系统，如 6.2.4 使用

LMI Shell 描述的。如果您打算直接在管理系统上执行 OpenLMI 操作，按照 6.2.2.2 在客户端安装 OpenLMI 中描述的步骤操作。

6.2.2.2. 在客户端安装 OpenLMI

客户端系统是可以让与您管理系统交互的系统。在通常情况下，客户端系统和管理系统都安装在两台不同的机器上，但您也可以在管理系统上安装客户端工具和管理系统直接互动。

在客户端系统安装 OpenLMI 按照如下步骤：

1) 以 root 身份安装 openlmi-tools 包：

```
yum install openlmi-tools
```

这个命令安装了 LMIShell，它是一个交互式客户端和解释器用来访问由 OpenPegasus 提供的 CIM 对象，及其所有依赖。

2) 为 OpenPegasus 配置 SSL 证书如 6.2.3 为 OpenPegasus 配置 SSL 证书描述的。

6.2.3. 为 OpenPegasus 配置 SSL 证书

OpenLMI 使用基于 Web 的企业管理（WBEM）协议，其功能通过 HTTP 传输层。在这个协议中执行标准的 HTTP 基本认证，这意味着该用户名和密码和请求一同发送。

配置 OpenPegasus CIMOM 使用 HTTPS 进行通信以确保安全认证是必要的。管理系统上建立一个加密通道需要安全套接字层（SSL）或传输层安全（TLS）证书。

在系统中有两种方法管理 SSL/TLS 证书。

- 自签名证书需要较少的基础设施，但是更难以安全的部署到客户机和管理端。
- 机构签发的证书更容易部署到客户端，但可能需要更大的初始投资。

当使用机构签发的证书，就必须在客户端系统上配置受信任的证书颁发机构。机构可以被用来签署的所有管理型系统的 CIMOM 证书。证书也可以是一个证书链的一部分，所以用于签名的管理系统的证书可能又被另一个更高一级的主管部门（如 Verisign，CAcert，RSA 及其他）签名。

默认的证书和信任的存储位置在表格 6-5 证书和信任的存储位置。

表格 6-5 证书和信任的存储位置

配置选项	位置	描述
sslCertificateFilePath	/etc/Pegasus/server.pem	CIMOM 的公共证书

sslKeyFilePath	/etc/Pegasus/file.pem	CIMOM 的私有 KEY
sslTrustStore	/etc/Pegasus/client.pem	文件或目录提供了信任证书颁发机构的列表

重要说明：

如果您修改任何表格 6-5 证书和信任的存储位置提到的文件，重新启动 tog-pegasus 服务以确保它识别新证书。以 root 身份键入如下命令：

```
systemctl restart tog-pegasus.service
```

6.2.3.1. 管理自签名证书

自签名证书使用自己的私钥签名本身并没有连接到任何信任链。在一个管理系统中，如果 tog-pegasus 服务启动之前管理员没有提供证书，一套自签名的证书将自动产生，该证书使用该系统的主机名做证书标题。

重要说明：

自动生成的自签名证书在默认情况下 10 年有效，但他们没有自动更新功能。任何修改都需要由 OpenSSL 或 Mozilla NSS 相关官方文档中提供的操作指南来手动创建新的证书。

在客户系统上配置信任自签名证书按下面步骤操作：

1) 从管理系统拷贝/etc/Pegasus/server.Pem 证书到客户机的/etc/pki/catrust/source/anchors/下，以 root 用户在 shell 里执行如下命令：

```
scp root@ hostname: /etc/Pegasus/server.pem /etc/pki /ca-trust/source/anchors/pegasus-hostname.pem
```

替换管理系统的主机名。请注意，此命令只有 sshd 服务是在管理系统上运行才生效，管理系统被配置被允许 root 用户通过 SSH 协议登录。通过对比原文件和新文件的校验值确认文件的完整性。在管理系统上计算/etc/Pegasus/server.Pem 文件的校验值，以 root 身份运行如下命令：

```
sha1sum /etc/Pegasus/server.pem
```

在客户端校验/etc/pki/catrust/source/anchors/pegasus-hostname.Pem 文件输入如下命令：

```
sha1sum /etc/pki/ca-trust/source/anchors/pegasus-hostname.pem
```

用管理系统的主机名替代 hostname。

2) 在客户端更新信任存储需要以 root 用户运行如下命令：

```
update-ca-trust  extract
```

6.2.3.2. 管理机构签发的证书与身份管理(推荐)

中标麒麟高级服务器操作系统软件的身份管理功能提供了简化的 SSL 证书中加入系统管理的域控制器。其中，身份管理服务器提供了一个嵌入式证书颁发机构。

注册管理系统到身份管理是必要的;对客户端系统的注册是可选的。

在管理端执行如下步骤：

1) 安装 ipa-client 包。

2) 拷贝身份管理签名到信任的存储系统，以 root 身份执行如下命令：

```
cp  /etc/ipa/ca.crt  /etc/pki/ca-trust/source/anchors/ipa.crt
```

3) 更新存储以 root 身份执行如下命令：

```
update-ca-trust  extract
```

4) 在身份管理域注册 Pegasus 服务，以特权域用户执行如下命令：

```
ipaservice-add  CIMOM/hostname
```

用管理系统的主机名替代 hostname.

这个命令可以从安装了 ipa-admintools 包中的身份管理域中的任何系统上运行。它创建一个可用于生成签名的 SSL 证书身份管理服务入口。

5) 备份本地的 PEM 文件到/etc/Pegasus/目录（推荐）。

6) 以 root 用户执行以下命令检索签名证书：

```
ipa-getcert  request  -f  /etc/Pegasus/server.pem  -k
/etc/Pegasus/file.pem  -N  CN=  hostname  -K  CIMOM/hostname
```

用管理系统的主机名替代 hostname。

证书和密钥文件现在保存在适当的位置。通过 ipa-client-install 脚本安装 certmonger 守护进程来确保证书更新是必要的。

注册客户系统更新信任存储按如下步骤：

1) 安装 ipa-client 包。

2) 拷贝身份管理签名到信任的存储系统，以 root 身份执行如下命令：

```
cp  /etc/ipa/ca.crt  /etc/pki/ca-trust/source/anchors/ipa.crt
```

3) 更新存储以 root 身份执行如下命令：

```
update-ca-trust extract
```

如果客户端系统并未在身份管理注册，请完成以下步骤来更新信任存储。

1) 以 root 用户从任何一个加入到同一个身份管理域的系统拷贝/etc/ipa/ca.crt 文件到信任存储域的/etc/pki/ca-trust/so urce/anchors/目录下：

2) 以 root 用户执行如下命令更新信任存储域：

```
update-ca-trust extract
```

6.2.3.3. 手动管理机构签发的证书

使用其他机制管理机构签发的证书比使用身份管理机制需要更多的手动配置。

有必要以确保所有的客户端都信任将要签署的管理系统证书。

- 如果证书权限缺省可信，则不必执行任何特殊的步骤来完成此。
- 如果证书颁发机构默认是不信任的，证书必须在客户端由管理系统导入。

拷贝证书到信任的存储域需要以 root 身份执行如下命令：

```
cp /path/to/ca.crt /etc/pki/ca-trust/source/anchors/ca.crt
```

更新信任存储域需要以 root 身份执行如下命令：

```
update-ca-trust extract
```

在管理系统完成如下步骤：

1) 创建一个新的 SSL 配置文件/etc/Pegasus/ssl.cnf 存储证书的信息，文件的内容实例如下：

```
[ req ]
distinguished_name     = req_distinguished_name
prompt                 = no
[ req_distinguished_name ]
C                      = US
ST                     = Massachusetts
L                      = Westford
O                      = Fedora
OU                     = Fedora OpenLMI
CN                     = hostname
```

使用管理系统的域名替代 hostname。

2) 以 root 身份执行如下命令在管理系统生成私有密钥：

```
openssl genrsa -out /etc/Pegasus/file.pem 1024
```

3) 以 root 身份执行如下命令生成证书:

```
openssl req -config /etc/Pegasus/ssl .cnf -new -key
/etc/Pegasus/file.pem -out /etc/Pegasus/server.csr
```

4) 将文件/etc/Pegasus/server.csr 发送到证书颁发机构进行签名。提交的文件的详细过程取决于特定认证机构。
5) 当从证书颁发机构收到签名证书,将文件存储为/etc/Pegasus/server. Pem。
6) 拷贝可信机构的证书到 Pegasus 信任的存储域，确保 Pegasus 能够相信自己的证书需要以 root 身份运行如下命令：

```
cp /path/to/ca.crt/etc/Pegasus/client.pem
```

完成所有描述的步骤后，即信任签名授权的客户都能够成功地与受管服务器的 CIMOM 通讯。

重要说明：

不同于身份管理解决方案的是如果证书过期，需要更新，所有描述的手动步骤，必须再次进行。建议在过期之前续订证书。

6.2.4. 使用 LMI Shell

LMIShell 是一个交互式客户端同时其非交互式解释可用于访问由的 OpenPegasus CIMOM 提供的 CIM 对象。它是基于 Python 解释器，同时它还实现了附加的函数和类用来和 CIM 对象交互。

6.2.4.1. 启动，使用，退出 LMIShell

类似于 Python 解释器，您可以使用 LMIShell 无论是作为一个交互式客户端或作为非交互式解释器来完成 LMIShell 脚本。

以交互模式启动 LMIShell

以交互模式启动 LMIShell，运行 lmishell 命令，无需其他参数。

```
lmishell
```

默认情况下，当 LMIShell 试图建立与 CIMOM 的连接，它验证由证书颁发机构信任存储的服务器端证书。要禁用此验证，命令使用--noverify 或-n 选项。

```
lmishell --noverify
```

使用 Tab 键完成

在交互模式下运行时，LMIShell 解释器允许您按 Tab 键完成基本的编程结构和 CIM 对象，包括命名空间，类，方法和对象属性。

浏览历史

默认情况下，LMIShell 在交互提示下存储所有您输入的命令到~/.lmishell_hi

story 文件。这可以让您浏览交互模式下命令历史记录，在交互模式下重新使用这些命令而无需再次键入。向后浏览的命令历史记录，按【上箭头】键或【Ctrl + p】组合键。要向前浏览命令历史记录，按【下箭头】键或【Ctrl+ n】组合键。

LMIShell 还支持增量反向搜索。如果寻找特殊的命令历史记录，按【Ctrl+ r】某一线并开始键入命令的任何部分。 例如：

```
> (reverse-i-search)`co nnect': c = co nnect("server.example.com",
"pegasus")
```

清空历史命令记录使用 clear_history()函数。

```
clear_history()
```

您可以通过配置~/.lmishellrc 文件中的 history_leng 值修改记录历史命令文件的行数。此外，您可以通过修改这个配置文件中的 history_file 选项的值更改此文件的位置。例如，设置历史文件位置为~ /.lmishell_history 的位置和最大行数为 1000。

```
history_file = "~/.lmishell_history"
history_length = 1000
```

处理异常

默认情况下，LMIShell 解释器使用返回值处理所有的异常。要禁用此行为以处理代码中的所有异常，使用 use_exceptions（）函数：

```
use_exceptions()
```

要使能自动处理异常，如下方法：

```
use_exception(False)
```

通过修改~/.lmishellrc 配置文件中 use_exceptions 值您可以永久禁用异常处理。

```
use_exceptions = True
```

配置临时缓存

在默认配置下，LMIShell 连接对象使用一个临时的缓存存储 CIM 类名和 CIM 类以减少网络通信。要清除此临时缓存，使用 clear_cache（）方法，如下：

```
object_name.clear_cache()
```

用连接对象的名字代替 object_name。

要为特定的连接对象禁用临时缓存，请使用 use_cache（）方法，如下：

```
object_name.use_cache(False)
```

使能临时缓存：

```
object_name.use_cache(True)
```

通过修改~/.lmishellrc 配置文件中 **use_cache** 值您可以对连接对象永久禁用临时缓存。

```
use_cache = False
```

退出 LMIShell

要终止 LMIShell 解释器并返回到 shell 提示符下，按【Crtl + d】组合键或发出的 quit（）函数,如下:

```
> quit()
~]$
```

运行 LMIShe ll 脚本

运行 LMIShe ll 脚本，执行如下命令：

```
lmishell file_name
```

脚本名字代替 file_name，执行后检查的 LMIShell 脚本，可指定--interact 或 -i 命令行选项：

```
lmishell --interact file_name
```

LMIShell 脚本的首选文件扩展名是.lmi。

6.2.4.2. 连接 CIMOM

LMIShell 允许您连接到的 CIMOM，要么是在同一个系统上本地运行要么在通过网络可以访问远程计算机上。

连接远程 CIMOM

要访问远程 CIMOM 提供的 CIM 对象，通过使用 connect（）函数，如下所示创建一个连接对象：

```
connect(host_name, user_name[, password])
```

用管理系统的主机名代替 host_name，用在该系统上可以连接 OpenPegasus CIMOM 的用户名代替 user_name，用户的密码替代 password。如果省略了密码，LMIShell 提示用户输入密码。该函数返回一个 LMIConnection 对象。

6.2.4.3. 使用命令空间

LMIShell 命名空间提供了一种组织可用类的自然方法，并对其他的命名空间和类作为一个分层接入点。根命名空间是连接对象的第一个进入点。

列出可用的命名空间

要列出所有可用命名空间，请使用 print_namespaces（）方法，如下所示：

```
object_name.print_namespaces()
```

用对象名替代 object_name，这个方法会向标准输出打印出可用的命名空间。

获取命名空间列表，访问对象的命名空间属性。

```
object_name.namespaces
```

这个方法返回字符串列表。

6.2.5. 使用 OpenLMI 脚本

LMIShell 解释器建立在 Python 模块上，可以用来开发自定义管理工具。OpenLMI 脚本项目提供了大量的 Python 库与 OpenLMI 提供商接口。此外，它和 lmi 一同发布，lmi 是一个可扩展工具，可以被用来与来自命令行的库交互。

在系统上安装 OpenLMI 脚本，在 shell 下输入如下命令：

```
easy_install --user openlmi-scripts
```

此命令在~/.local/目录下安装 Python 模块和 lmi 工具。为了扩展 lmi 工具的功能，安装附加的 OpenLMI 模块，使用如下命令：

```
easy_install --user package_name
```

完整可用模块的列表请参考 Python 官方网站，更多关于 OpenLMI 脚本的信息参考 OpenLMI 官方脚本文档。

6.3. 查看和管理日志文件

日志文件是包含系统信息的文件，包括内核，服务及其上运行的应用程序。不同的日志文件包含不同的信息，例如，有一种默认系统日志文件，一种只是用于安全信息的日志文件，还有用于后台任务的日志文件。

当试图解决系统问题的时候，如试图载入内核驱动程序或寻找未授权的登录系统时，日志文件是非常有用的。本章讨论在哪里可以找到日志文件，如何查看日志文件，以及在日志文件中查看什么。

一些日志文件被一个名为 rsyslogd 的守护进程控制。该 rsyslogd 守护进程是替代 sysklogd 的一种增强型进程，并提供了扩展的过滤器，消息加密保护，各种

配置选项，输入输出模块，通过 TCP 或 UDP 协议进行传输。需要注意的是 rsyslogd 与 sysklogd 是兼容的。

日志文件还可以由 journald 守护进程进行管理，该守护进程是 systemd 的组件。journald 守护进程捕获系统日志消息，内核日志消息，初始 RAM 磁盘和早期启动消息和写到标准输出的消息，以及所有服务的标准错误输出，把这些消息进行索引，并提供给用户。本地日志文件格式，它是一种结构化和索引二进制文件，改进了搜索，并提供更快的操作，而且它也存储像时间戳或用户 ID 的元数据信息。由 journald 生成的日志文件在默认情况下不是一直存在的，日志文件只保存在内存或/run/log/journal/目录下的环形缓冲区。记录的数据量取决于可用内存，当您达到容量极限时，最早的记录将被删除。但是，设置可以被改变 - 请参阅 6.6.5 使能持续存储。有关期刊详细信息，请参阅 6.6 使用日志。

默认情况下，这两个记录工具并存在您的系统。journald 守护进程是解决问题的主要工具。它也提供了必要的用于创建结构化日志消息的附加数据。通过 journald 获取的数据被转发到/run/systemd/journal/syslog 套接字，可以由 rsyslogd 进一步处理数据。然而，rsyslogd 默认通过 imjournal 输入模块，从而避免了上述的套接字。您还可以使用 omjournal 模块从 rsyslogd 到 journald 在相反方向上传输数据。参见 6.3.7 Syslogd 服务和日志的交互。集成使基于文本的日志文件使用一致的格式方便用来维护，以确保可能的应用程序或配置依赖于 rsyslogd 的兼容性。另外，您可以用结构一致化的格式维护 rsyslogd 日志。（参见 6.4Syslogd 日志结构）

6.3.1. 日志文件的位置

许多由 rsyslogd 维护的日志文件在/etc/rsyslog.conf 配置文件里记录。大多数的日志文件在/var/log/目录下。一些应用程序例如 httpd 和 samba 在/var/log/目录下有自己的日志文件。

您可能会注意到在/var/log 目录下有多个日志文件其名字后的数字不同（例如，cron-20100906）。这些数字代表添加轮替日志文件里的时间戳。日志文件被轮替，所以其大小不会太大。Logrotate 包括了一个后台任务，它可以根据/etc/logrotate.conf 配置来自动轮替日志文件，该配置文件在/etc/logrotate.d/目录下。

6.3.2. Rsyslog 的基本配置

Rsyslog 的主要配置文件是/etc/rsyslog.conf。在这里，您可以指定全局指令，模块和过滤器的规则和动作部分。此外，您还可以以文本形式评论添加解释（以#号开头）。

6.3.2.1. 过滤器

规则是过滤器的一部分，它选择系统日志消息的一个子集，以及行动的一部

分，它指定对选中的消息做什么。要在您的/etc/rsyslog.conf 的配置文件定义规则，在一行同时定义过滤器和动作，用一个或多个空格或制表符分隔。

根据所选属性，rsyslog 现在提供了不同的方式过滤系统日志消息。可用的过滤方法，可分为设置/基于优先级的，基于属性的，和基于正则表达式的过滤器。

设置/基于优先级的过滤器

最常用的和众所周知的方式来过滤系统日志消息是使用基于设置/基于优先级的过滤器，其过滤 syslog 消息基于两个条件：由点号分隔设置和优先级。要创建一个选择器，请使用以下语法

```
FACILITY.PRIORITY
```

- FACILITY 指定一个特定的子系统来生成日志消息。例如，邮件子系统处理所有与邮件相关的系统日志消息。FACILITY 可以由以下关键字之一（或由数字代码）来表示：kern (0), user (1), mail (2), daemon (3), auth (4), syslog (5), lpr (6), news (7), uucp (8), cron (9), authpriv (10), ftp (11), 和 local0 through local7 (16 - 23).
- PRIORITY 指定系统日志消息的优先级。PRIORITY 可以由以下关键字之一（或由数字代码）来表示：debug (7), info (6), notice (5), warning (4), err (3), crit (2), alert (1), 和 emerg (0).

上述语法根据定义或更高优先级选择系统日志消息。通过比较等号（=）两边的优先级，您指定的优先级的系统日志消息将被选中，所有其他优先级将被忽略。相反，优先关键字之前有一个（!），选择除了该优先级的所有系统日志消息。

除上述指定的关键字，您也可以使用星号（*）来定义所有设置或优先级（这取决于您在哪里放置星号，逗号之前或之后）。没有给出优先级设备的优先级指定其关键字为 none。设置和优先条件是不区分大小写。

要定义多个设置和优先级，用逗号（,）将它们分开。要在一行中定义多个选择，用分号（;）分隔它们。注意，在选择器在其字段能够覆盖前面的部分，它可以排除来自模式一些优先次序。

实例：设置/基于优先级的过滤器

以下是可在/etc/rsyslog.conf 中指定设置/基于优先级的过滤器的几个简单例子。要选择所有优先级的所有内核系统日志消息，添加下面的文字到配置文件：

```
kern.*
```

选择所有的邮件系统日志消息，使用优先级为 crit 或更高，使用如下形式：

```
mail.crit
```

选择所有的 cron 日志消息除了优先级为 info 和 debug 的。使用如下格式：

```
cron.!info,!debug
```

基于属性的过滤器

基于属性的过滤器可让您过滤系统日志消息的任何属性，如 timegenerated 或 syslogtag。您可以将每个指定属性和值比较，这些值使用表格 6-6 基于属性的比较操作 .列出的比较操作。属性名和比较操作都是区分大小写的。

基于属性的过滤器以冒号（:）开始，定义该过滤器，使用如下语法：

```
:PROPERTY, [!]COMPARE_OPERATION, "STRING"
```

- PROPERTY 定义需要过滤的属性。
- 可选的感叹号（!）反向输出比较操作。基于属性的过滤器目前不支持其他布尔运算符。
- COMPARE_OPERATION 属性定义了比较操作，参考表格 6-6 基于属性的比较操作基于属性的比较操作 .
- 字符串属性指定比较值，其由属性的文字字符串提供。该值必须用引号括起来。为了使用某些字符的字符串中（例如一个引号 (“)），用反斜杠字符 (\)。

表格 6-6 基于属性的比较操作

Compare-operation	description
Contains	检查提供的字符串是否有任何部分和所提供的文本属性字符串相匹配。要执行大小写敏感的比较，使用 contains_i。
isequal	比较提供的字符串和文本提供的属性字符串。这两个值必须匹配。
startswith	检查提供字符串和文本的属性字符串开头的匹配。要执行不区分大小写的比较，使用 startswith_i。
regex	比较提供的 POSIX BRE（基本正则表达式）和文本所提供的属性字符串。
ereregex	比较提供的 POSIX ERE（扩展正则表达式）和文本所提供的属性字符串。
isempty	检查该属性是否为空。值被丢弃。在使用归一化的数

	据时，某一些字段可能被填充，这个操作则特别有用。

实例：基于属性的过滤器

以下是可在/etc/rsyslog.conf 指定基于属性的过滤器的几个例子。在消息文本里要选择包含 error 字符串的日志，如下所示：

```
:msg, contains, "error"
```

下面的过滤器从主机名为 host1 条件选择日志消息。

```
:hostname, isequal, "host1"
```

选择的日志消息不包含 fatal 和 error 已经其之间的文本。

```
:msg, !regex, "fatal .* error"
```

基于表达式的过滤器

基于表达式过滤器根据定义的算术，布尔或字符串操作选择系统日志消息。基于表达式过滤器使用 rsyslog 自己的脚本语言调用 RainerScript 脚本，可以构建复杂的过滤器。

基本的基于表达式的过滤器使用如下：

```
if EXPRESSION then ACTION else ACTION
```

- EXPRESSION 属性表示要计算的表达式。例如：下面的表达式

$msg startswi th ' DEVNAME' or $syslogfacility-text = = ' local0'您可以定义多个表达式，使用 and 或者 or 操作符。

- ACTION 属性代表如果表达式返回为真要执行的动作。这可以是一个单一的动作，或包含在大括号的任意复杂的脚本。
- 在一个新行的开始，基于表达式过滤器由关键字 if 指示。关键字 then 分开 EXPRESSION 和 ACTION。或者，也可以用关键字 else 来指定哪些动作是如果条件得不到满足将要执行。

基于表达式的过滤器，可以使用大括号嵌套使用条件就像例如：基于表达式的过滤器。该脚本可以让您在表达式中使用设备/基于优先级的过滤器。另一方面，基于属性的过滤器，不建议在这里。RainerScript 支持特定函数 re_match（）和 re_extract（）的正则表达式。

实例：基于表达式的过滤器

下面的表达式包含两个嵌套条件。prog1 程序创建的日志文件被分成基于消息中的“test”字符串的两个文件。

```
if $programname == 'prog1' then {
   action(type="omfile" file="/var/log/prog1.log")
   if $msg contains 'test' then
     action(type="omfile" file="/var/log/prog1test.log")
   else
     action(type="omfile" file="/var/log/prog1notest.log")
}
```

更多的关于基于表达式的过滤器参考在线文档。RainerScript 脚本是 rsyslog 新配置格式的基础，请参考 6.3.3 使用新的配置格式。

6.3.2.2. 行为

行为指定对已经过滤的消息做什么。下面是一些可以在您的规则中定义的操作：

存储 syslog 消息到日志文件

主要行为指明 syslog 消息被保存哪个日志文件。这是由您指定文件路径完成。

```
FILTER  PATH
```

FILTER 代表着用户选择的目标文件的路径。例如，下面的规则会选择所有的 corn syslog 消息然后将其存储到/var/log/cron 文件中。

```
cron.*  /var/log/cron
```

默认情况下，日志文件每次生成一个系统日志消息的时间同步。使用破折号标记（ - ）作为文件路径的前缀指定省略同步：

```
FILTER  -PATH
```

请注意，您可能会失去信息，如果系统终止发生在写之前。但是，这种设置可以提高性能，特别是如果您运行的程序会产生非常详细的日志信息。

您指定的文件路径可以是静态或动态的。静态文件由一个固定文件路径，正如上所示的例子中表示的。动态文件路径根据所接收的消息是不同的。动态文件路径是由一个问号（？）的前缀代表表示：

```
FILTER  ?DynamicFile
```

DynamicFile 是一个名称。相当于预定义的模板其用于修改输出路径。您可以使用破折号做前缀（ - ）来禁用同步，您也可以使用一个冒号分隔的多个模板（;）。

当您使用的是 X Window 系统的时候，如果您指定的文件是存在的终端或/dev/console 设备，系统日志消息发送到标准输出（使用特殊的终端处理）或控制

台（使用特殊的/dev/console 的-处理）。

通过网络发生syslog消息

Rsyslog 允许您可以通过网络发送和接收系统日志消息。此功能允许您在一台机器管理多个主机的系统日志消息。要转发 syslog 消息到远程计算机，请使用以下语法：

```
@ [(zNUMBER)]HOST:[PORT]
```

- at 符号（@）表示 syslog 消息被转发到使用 UDP 协议的主机。要使用 TCP 协议，使用两个 at 符号与他们之间没有空格（@@）。
- 可选项 zNUMBER 设置对系统日志消息的 zlib 压缩。该 NUMBER 属性指定的压缩级别（从 1 - 最低到 9 - 最大）。压缩增益自动由 rsyslogd 检查，如果有任何压缩增益消息会被压缩，如果消息小于 60 字节则不会被压缩。
- HOST 属性定义哪个主机选择 syslog 消息。
- PORT 属性定义主机端口。
- 当主机定义 IPV6 地址，用方括号括地址（[]）。

实例：通过网络发送 syslog 消息

以下是该通过网络（请注意，所有操作都前面带有一个选择器，其选择任何优先级的所有消息）转发系统日志信息的操作的一些例子。要通过 UDP 协议，将消息转发到 192.168.0.1。

```
*.* @ 192.168.0.1
```

使用 6514 端口和 TCP 协议将消息转发到 example.com：

```
*.* @ @ example.com:6514
```

下面压缩信息以 zlib（9 级压缩），并将其转发给[2001:db8::1],传输使用 UDP 协议。

```
*.* @ (z9)[2001:db8::1]
```

输出通道

输出通道主要用于指定日志文件可以增长到的最大大小。这对日志文件轮替非常有用的（有关详细信息，请参见 6.3.2.4 日志轮替）。输出通道基本上是关于输出操作的信息的集合。输出通道由$outchannel 指令定义。要定义/etc/rsyslog.conf 输出通道，请使用以下语法：

```
$outchannel NAME, FILE_NAME, MAX_SIZE, ACTION
```

- NAME 属性指定了输出通道的名称。
- FILE_NAME 属性指定了输出文件名。输出通道只能写入到文件中，没有管道，终端，或其他类型的输出。
- MAX_SIZE 属性指定了文件可以增长的最大大小，以字节为单位。
- ACTION 属性定义当文件到最大大小的行为。
- 要使用定义的通道作为规则里的一个行为，如下：

```
FILTER :omfile:$NAME
```

实例：输出通道日志轮替

下面通过使用一个输出通道显示了一个简单的日志轮替。输出通道经由$outchannel 指令定义：

```
$outchannel log_rotation, /var/log/test_log.log, 104857600,
/home/joe/log_rotation_script
```

使用规则选择所有优先级的所有 syslog 消息，获取消息后执行预先定义的输出通道。

```
*.* :omfile:$log_rotation
```

一旦限制（在本例中 100 MB）被达到， /home/joe/ log_rotation_script 会被执行。该脚本可以包含任何从文件移动到其他文件夹内容，编辑具体的内容，或者干脆删除它。

发送 syslog 消息给特定的用户

rsyslog 通过指定用户名将 syslog 消息发送到指定的用户（如本章实例：指定多行为）。要指定多个用户，用逗号（,）分隔每个用户名。将消息发送给当前登录的所有用户，使用星号（*）。

执行一个程序

rsyslog 可以为选定的系统日志消息执行程序，并使用 system（）调用在 shell 中执行的程序。要指定一个程序来执行，用（^）字符前缀该命令。因此，指定一个模板其接收的消息的格式，并将其传递到指定的可执行文件作为一个行参数。

```
FILTER ^EXECUTABLE; TEMPLATE
```

这里过滤器状态的输出通过由 EXECUTABLE 指定的程序进行处理。这个程序可以是任何有效的可执行文件。用格式模板的名称替换 TEMPLATE。

实例：执行程序

在下面的例子中，被选择的任何优先级的任何系统日志消息，其格式经过与 template 模板匹配并作为参数传递给 test-program 程序。

```
*.* ^test-program;template
```

警告：

从任何主机接收消息时以及使用 shell 执行操作，可能会受到命令注入。攻击者可以尝试将命令注入到您指定的行为要执行的程序。为了避免任何可能的安全威胁，充分考虑使用的 shell 执行行为。

存储 syslog 消息到数据库

通过使用数据库写操作，被选择的系统日志消息可以直接写入到数据库表。数据库写入使用的语法如下：

```
:PLUGIN:DB_HOST,DB_NAME,DB_USER,DB_PASSWORD;[TEMPLATE]
```

- PLUGIN 调用特定的插件，这些插件可以处理数据库写操作。（例如，ommysql 插件）。
- DB_HOST 属性指定数据库的主机名。
- DB_NAME 属性指定数据库名。
- DB_USER 属性指定数据库用户。
- DB_PASSWORD 属性指定数据库用户的密码。
- TEMPLATE 属性指定一个修改 syslog 信息的模板。

重要：

目前，rsyslog 只支持 MySQL 和 PostgreSQL 数据库。为了使用 MySQL 和 PostgreSQL 数据库写入功能，安装 rsyslog-mysql 和 rsyslog-pgsql 包。此外，请确保您在您的/etc/rsyslog.conf 配置文件加载相应的模块：

```
$ModLoad ommysql # Output module for MySQL support
$ModLoad ompgsql # Output module for PostgreSQL support
```

更多关于 rsyslog 的信息参考 6.3.6 使用 Rsyslog 模块。

另外，您也可以使用由 omlibdb 模块提供通用的数据库接口（支持：Firebird/Interbase, MS SQL, Sybase, SQLLite, Ingres, Oracle,mSQL）。

丢弃 syslog 消息

想丢弃选择好的消息使用波形符（～）.

```
FILTER ~
```

丢弃行为主要是用来在进行任何进一步的处理之前，以筛选出的消息。如果

您想省略一些重复的消息，这是非常有效的，否则重复消息将填充日志文件。丢弃操作的结果取决于所在的配置文件中的指定配置的位置，为得到最好的结果把这些行为放到行为列表的前面。请注意，一旦消息被丢弃在后面的配置文件中的行则没有办法来检索它。

例如，下面的规则丢弃了任何 cron 的 syslog 消息。

```
cron.* ~
```

定义多行为

对于每一个选择，您被允许指定多个行为。对一个选择要指定多个操作，每一个行为写在单独一行，并用符号它前面（&）字符：

```
FILTER ACTION
& ACTION
& ACTION
```

指定多个操作提高了所希望结果的整体性能，因此选择器需要被重新评估。

6.3.2.3. 全局指令

全局指令是适用于 rsyslogd 守护进程的配置选项。他们通常会指定特定预定义变量的值，该值会影响 rsyslogd 守护进程的行为或遵循的规则。所有全局指令必须以美元符号（$）。只有一个指令是每行指定。下面是一个全局指令的例子，其指定系统日志消息队列的最大大小：

```
$MainMsgQueueSize 50000
```

这个指令默认大小是 10000 个消息，它可以被上面的例子重新不同的数字。

您可以在/etc/rsyslog.conf 配置文件中定义多个指令。一个指令影响的所有配置选项的行为，直到同一个指令的另一事件被检测到。全局指令可以用来配置行为，队列和调试。所有可用的配置指令的完整列表可以参考在线文档。目前，一种新的配置格式已经开发出其可以替代$基础的语法（6.3.3 使用新的配置格式）。然而，经典的全局指令仍然为旧格式的支持。

6.3.2.4. 日志轮替

下面是一个简单的/etc/logrotate.conf 配置文件例子：

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# uncomment this if you want your log files compressed
```

```
compress
```

示例配置文件中所有的行的定义全局选项，其适用于每一个日志文件。在我们的例子，日志文件每周轮替，轮替的日志文件保留四个星期，所有的轮替日志文件由 gzip 压缩的成.gz 的格式。以#号开头的任何行是注释，不会被处理。

您可以为特定的日志文件定义配置选项，并将其放置在全局选项的下面。然而，最好是在/etc/logrotate.d/directory 目录下为特定的日志文件创建单独的配置文件，并在配置文件里配置选项。

下面是一个在/etc/logrotate.d/目录下配置文件的例子：

```
/var/log/messages {
rotate 5
weekly
postrotate
/usr/bin/killall -HUP syslogd
endscript
}
```

该文件中的配置选项只为/var/log/messages 日志文件配置。在可能的情况下，在此指定的设置将覆盖全局设置。因此，轮替的/var/log/messages 日志文件将保留五周，而不是全局选项定义定义的四周。

下面是一些指令的列表，您可以在您的日志轮替配置文件里修改。

➢ weekly- 指定日志轮替的周期为每周，类似的命令还包括如下：

daily

monthly

yearly

➢ compress-使能轮替文件的压缩功能，类似的命令还包括如下：

nocompress

compresscmd – 指定用于压缩的命令。

uncompresscmd

compressext – 指定用于压缩的扩展。

compressopttions – 指定用于传给程序的压缩选项。

delaycompress – 退出压缩直到下一个轮替周期。

➢ rotate INTEGER -指定了一个日志文件被轮替的最大数目，超过数目的日志文件被删除或邮寄到一个特定的地址。如果指定值 0，旧的日志文件被删除，而不会轮替。

➢ mail ADDRESS -此选项，已轮替多次的日志文件的邮件地址。类似的指令包括：

nomail

maifirst-指定只是轮替的日志文件被邮寄，到期的日志文件不被邮寄。

Maillast- 指定到期的日志文件被邮寄，轮替的日志文件不邮寄。当邮寄功能打开时，此选项是默认选项。

关于个配置选项的完整的指令列表参考 logrotata(5)手册页。

6.3.3. 使用新的配置格式

中标麒麟高级服务器操作系统软件安装的 rsyslog 包默认情况下是第 8 版，介绍其新的配置语法。这种新的配置格式的目标是更强大，更直观，通过不允许某些无效的结构防止常见的错误。语法增强的使能是通过 RainerScript 脚本配置处理器完成。遗留格式仍然得到支持，它默认在/etc/rsyslog.conf 配置文件中使用。

RainerScript 是一种脚本语言，旨在用于处理网络事件和配置事件处理器，例如 rsyslog。RainerScript 最早是用来定义基于表达式的过滤器，见 6.3.2.1 过滤器中有关基于表达式过滤器的章节。RainerScript 在 rsyslog7 的版本中实现了 input（）和 ruleset（），其允许所述/etc/rsyslog.conf 配置文件被写入新的语法。新的语法不同之处主要在于，它是更结构化的;参数作为参数传递给语句，如输入，行为，模板，模块加载。选项的范围由块限制。这增强可读性，并降低所造成由错误配置的错误的数量。还有一个显著的性能优势。有些功能在新旧语法都可以用，有的只在新语法可以用。

与旧风格的参数配置比较：

```
$InputFileName /tmp/inputfile
$InputFileTag tag1:
$InputFileStateFile inputfile-state
$InputRunFileMonitor
```

同样的配置使用新的配置语句如下：

```
input(type="imfile" file="/tmp/inputfile" tag="tag1:"
statefile="inputfile-state")
```

这显著减少在配置中使用的参数的数目，提高了可读性，并且还提供了更高的运行速度。有关 RainerScript 语句和参数的详细信息，请参见在线文档。

6.3.3.1. 规则集

除特殊指令之外，rsyslog 处理的消息由规则定义。规则由过滤条件和如果条件为真要执行的操作组成。在/etc/rsyslog.conf 文件中的旧规则，所有的规则以每个输入消息的出场顺序来评估。此过程开始于第一规则，并继续，直到所有的规

则都已经被处理或直到消息被其中某一规则丢弃。

然而，规则可被分成不同序列称为规则集。在规则集中，您可以限制某些规则的影响，只选择输入或通过定义一组绑定到特定的输入的行为提升 rsyslog 的性能。换句话说，某些类型的消息不可避免的被评估为假的，类似这样的过滤条件可以被跳过。在/etc/rsyslog.conf 中旧规则集定义如下所示：

```
$RuleSet rulesetname
rule
rule2
```

当另一个规则被定义，上一个规则就结束了，或者默认规则集被调用，如下：

```
$RuleSet RSYSLOG_DefaultRuleset
```

rsyslog 8 的新的配置格式，input（）和 ruleset（）语句执行被保留。新格式的规则集在/etc/rsyslog.conf 中定义，如下所示：

```
ruleset(name="rulesetname") {
rule
rule2
call rulesetname2
…
}
```

用您规则集的标识符替换 rulesetname。该规则集名称不能以 RSYSLOG_开始，因为这个名称空间是保留的，供 rsyslog 使用。RSYSLOG_DefaultRuleset 定义缺省设置的规则集，如果消息没有分配其他规则集将被执行。在上面提到的 rule 和 rule2 下，您可以定义过滤-操作的格式规则。对于调用参数，您可以嵌套规则集由内部或者规则集块调用它们。

创建规则集后，您需要指定它适用于什么输入：

```
input(type="input_type" port="port_num" ruleset="rulesetname");
```

这里可以通过 input_type 识别输入消息，这是所收集的信息，或者通过 port_num - 端口号。其他参数，如 file 或 tag 可以用于指定 input（）。用规则集的名称替换 rulesetname。万一某个输入消息未明确在规则集中指定，默认规则集会被触发。

6.3.3.2. sysklogd 服务的兼容性

rsyslog 第 5 版通过-c 选项指定兼容模式，但在第 8 版不支持，同样， sysklogd 风格的命令行选项已过时，应避免通过这些命令行选项配置 rsyslog。但是，

您可以使用多个模板和指令配置 rsyslogd 来模仿像 sysklogd 似的行为。

更多关于不同的 rsyslogd 选项的信息参考 rsyslogd（8）手册页。

6.3.4. 使用 rsyslog 队列

队列用于在 rsyslog 的组件之间传递内容，传递的主要内容是系统日志消息。在队列机制下，rsyslog 能够同时处理多个消息，并可以用多个行为对单个消息立刻反应。rsyslog 的数据流说明如下：

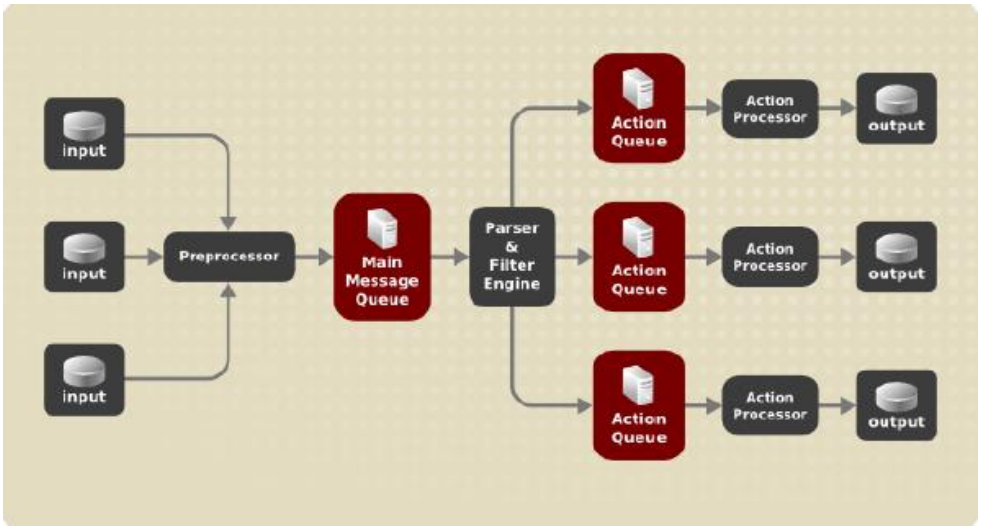


图 6-4 rsyslog 中消息流

无论 rsyslog 什么时候收到消息，它都把这个消息给预处理，然后将其放置到主消息队列。消息等在那里待出队并传递到规则处理器。

规则处理器是一个解析和过滤引擎。在此，在/etc/rsyslog.conf 定义的规则被应用。基于这些规则，该规则处理器评估哪些操作被执行。每个行为都有它自己的执行队列。消息通过队列传递到行为处理器最终产生相应的输出。注意，在这一点上，几个行为可以同时对一个消息运行。为了这个目的，一个消息被复制并传递到多个行为处理器。

只有每个行为一个队列是可能的。根据配置的不同，该消息可以正确发送到行为处理器而不通过执行队列。这是直接队列（见下文）的行为。在输出操作失败的情况下，行为处理器通知执行队列，一段时间间隔后，行为处理器又会去取未处理 消息，再次尝试。

综上所述，rsyslog 队列有两个位置：无论是对与规则处理器作为一个主消息队列的前面或在各种类型的输出行为作为动作队列的前面。队列有两个主要优点既都会提高消息处理性能：

- 它们充当缓冲器，在 rsyslog 的结构中分离生产者和消费者。
- 它们允许并行执行消息。

除此之外，队列可以用几种不同的指令被配置来为系统提供最佳的性能。这些配置选项在以下各节介绍。

警告：

如果输出插件无法传送一个消息，它被存储在前面的消息队列。如果在队列满时，输入阻塞，直到它不再满。这将阻止新的消息进入被阻塞的消息队列。在没有单独的执行队列的情况下，这会产生严重的后果，例如阻止 SSH 日志记录，这反过来又阻止 SSH 访问。因此，建议为那些通过网络转发到数据库的输出使用专用的行为的队列。

6.3.4.1. 定义队列

根据消息在哪里存储，分为几种类型的队列：direct，in-memory，disk 和 disk-assisted,最广泛使用的队列是 in-memory。您可以选择这些类型的其中之一作为主要消息队列，也可以作为执行队列。将下面的语句添加到/etc/rsyslog.conf 文件：

```
$objectQueueType  queue_type
```

在这里，您可以为主消息队列（用 MainMsg 替换 object）或执行队列（用 action 代替 object）申请设置。用 direct，linkedlist 或 fixedarray（in-memory 队列），或 disk 更换 queue_type。

默认设置主消息队列是 FixedArray 队列,最大存储 10000 个消息。执行队列默认设置为 Direct 队列。

Direct 队列

对于许多简单的操作，例如，输出写入到本地文件，在执行前建立一个队列是不需要的。为了避免排队，使用如下：

```
$objectQueueType  Direct
```

使用 MainMsg 或者 Action 替代 object,这个选项可以用在主消息队列或者执行消息队列。对于 direct 队列，消息直接传送，它会很快从生产者传送到消费者。

Disk 队列

disk 队列存储消息到硬盘驱动器，这使得它们高度可靠的，但也是所有可能的排队模式中最慢的。此模式可用于防止高度重要的日志数据的丢失。但是，大多数用例中不建议使用 disk 队列。要设置 disk 队列，在/etc/rsyslog.conf 中键入以下内容：

```
$objectQueueType  Disk
```

使用 MainMsg 或者 Action 替代 object,这个选项可以用在主消息队列或者执行消息队列。disk 队列写入部分，默认大小为 10 MB。此默认大小可以通过下面的配置指令进行修改：

```
$objectQueueMaxFileSize size
```

其中，size 表示 disk 队列的大小。定义的大小限制不是强制性的，rsyslog 总是写入一个完整的队列条目，即使违反的大小限制。disk 队列的每个部分有单独的文件相匹配。这些文件的命名指令如下所示：

```
$objectQueueFilename name
```

这为要设置的文件设置了文件名前缀。其文件名以 7 个数字开始，每增加一个文件数字也增加。

In-memory 队列

在 in-memory 队列中，排队的消息被保存在内存中，这使得这个过程非常快。如果电脑重新加电或关机,则排队的数据将丢失。但是，您可以使用$ActionQueueSaveOnShutdown 设置关机前保存数据。有两种类型的 in-memory 队列：

- FixedArray queue –默认的主要消息队列，最大 10,000 个元素。这种类型的队列的使用固定预分配的数组保存队列元素的指针。由于这些指针，即使队列为空也会消耗一定量的存储器。然而，FixedArray 提供了最佳的运行时性能，当您期望排队的消息相对较少和高性能，此队列是最佳的。
- LinkedList queue –这里，一个链表里所有结构都是动态分配的，因此，存储器仅在需要时分配。LinkedList 的队列处理偶尔的突发消息非常好。

在一般情况下，当使用 LinkedList 队列相比于 FixedArray 队列，它消耗更少的内存，并降低了处理开销。

配置 in-memory 队列使用下面的语法：

```
$objectQueueType LinkedList
$objectQueueType FixedArray
```

使用 MainMsg 或者 Action 替代 object,这个选项可以用在主消息队列或者执行消息队列。

disk-assisted in-memory 队列

disk 和 in-memory 队列都有各自的优点,rsyslog 让您可以将其合并为 disk-assisted in-memory 队列。为此，配置一个正常的 in-memory 队列然后添加$bjectQueueFileName 指令可定义为磁盘援助的文件名。这个队列就变成 disk-assisted，这意味着 in-memory 队列与 disk 队列通力协作。

如果 in-memory 队列已满或者需要关机后消息不丢的 disk 队列被激活。在 disk-assisted 队列中，可以同时设置 disk-specific 或 in-memory specific 配置参数。这种类型的队列可能是最常用的，它对可能长时间运行的和不可靠的操作特别有用。

指定 disk-assisted in-memory 队列使用 so-called 水印：

```
$objectQueueHighWatermark number
$objectQueueLowWatermark number
```

使用 MainMsg 或者 Action 替代 object,这个选项可以用在主消息队列或者执行消息队列。使用允许入队的最大消息数替代 number。当 in-memory 队列达到了高水位定义的数字，它开始将消息写入磁盘，并一直持续到 in-memory 队列大小下降到与低水印定义的数量。正确设置水印尽量减少不必要的磁盘写操作，同时也留下了消息的内存空间，因为写入磁盘文件是相当慢的。因此，高水位必须比整个队列容量设置$objectQueueSize 低。高水位和整体队列大小之间的差是专供消息突发而备用的存储器缓冲区。在另一方面，设定高水位过低会不必要的频繁的打开磁盘援助。

6.3.4.2. 管理队列

所有类型的队列可以进一步配置以满足您的要求。您可以使用几种不同的指令来修改执行队列和主消息队列。目前，有 20 多个队列参数可用，见在线文档。其中的一些设置是常见的，其他诸如工作线程管理，提供对队列行为多的控制，保留给高级用户使用。在高级设置中，您可以优化 rsyslog 的性能，调度排队，或修改系统关闭队列的行为。

限制队列大小

您可以限制队列可以包含消息的数量，使用如下设置：

```
$objectQueueHighWatermark number
```

使用 MainMsg 或者 Action 替代 object,这个选项可以用在主消息队列或者执行消息队列。用队列可以包含的消息数目替代 number。可以设置队列的大小，而不是作为它们实际存储器的大小。主要消息队列和规则集队列默认队列大小为 10000，执行队列默认大小为 1000。

Disk assisted 队列在默认情况下是无限制的，不能用指令来强制大小，但可以以字节为单位保留他们的物理磁盘空间，使用以下设置：

```
$objectQueueMaxDiscSpace number
```

使用 MainMsg 或者 Action 替代 object。当队列被填满时，新消息被丢弃，

直到队列释放了足够的空间。

丢弃消息

当队列消息达到一定的数量，为了节省空间，您可以在队列中丢弃不太重要的信息，保留空间给优先级更高的消息。启动丢弃处理的阈值可设定 discard mark：

```
$objectQueueDiscardMark number
```

使用 MainMsg 或者 Action 替代 object,这个选项可以用在主消息队列或者执行消息队列。这里，number 表示要启动丢弃消息进程的消息数目。定义哪些消息要被丢弃，使用如下：

```
$objectQueueDiscardSeverity priority
```

用下面的关键字之一（或一些）替代 priority：debug（7），info（6），notice（5），warning（4），err（3），crit（2），alert（1）和 emerg（0）。使用此设置，当达到丢弃数目后，比定义优先级较低的新来的和已排队消息从队列中删除。

使用时限

可以配置 rsyslog 在一个特定的时间段处理队列。有了这个选项，您可以转移部分工作到非高峰时段。要定义一个时间范围，请使用以下语法：

```
$objectQueueDequeueTimeBegin hour
$objectQueueDequeueTimeEnd hour
```

您可以指定绑定时间的框架，以小时为单位。使用 24 小时制，不支持设置分。

配置工作线程

工作线程执行消息入队的指定操作。例如，在主消息队列，一个工作线程的任务将滤波器逻辑应用到每个输入消息，并将其排队到相关执行队列。当消息到达时，一个工作线程将自动启动。当消息的数量达到一定数量时，另一个工作线程被启动。要指定此数，使用如下：

```
$objectQueueWorkerThreadMinimumMessages number
```

用消息的数量替代 number 将触发补充工作线程。例如，number 置为 100，当 100 多个消息到达时，一个新的工作线程开始。当超过 200 个消息到达时，第三个工作线程启动等。然而，并行运行太多的工作线程是低效的，所以您可以限制它们的最大数量，如下所示：

```
$objectQueueWorkerThreads number
```

number 代表可并行运行的工作线程的最大数目。对于主消息队列，默认限制为 1 个线程。一旦一个线程工作已经开始，它会一直运行直到超时出现。要设置超时时长，如下所示：

```
$objectQueueWorkerTimeoutThreadShutdown time
```

用持续时间（以毫秒为单位）替换 time。如果没有设置 time，零超时会被应用，当它没有消息可以处理是，工作线程立即终止。如果您指定的时间为-1，没有线程将被关闭。

批量出队列

为了提高性能，您可以配置 rsyslog 同时多队列的消息数。设定出队列消息数目的上限，则使用如下命令：

```
$objectQueueDequeueBatchSize number
```

用可被同时出队列的消息的最大数目替换 number。请注意，出队列消息数目较高的设置和很多的工作线程同时运行会导致较大的内存消耗高。

终止队列

当要终止的队列中仍然包含信息，您可以尝试通过指定时间间隔使工作线程来完成队列处理，以减少数据丢失：

```
$objectQueueTimeoutShutdown time
```

指定的 time 以毫秒为单位。如果一段时间后仍然有一些排队的消息，工作线程完成当前的数据处理，然后终止队列。因此，未处理的消息都将丢失。另一个时间间隔可以让工作线程完成最后的数据处理：

```
$objectQueueTimeoutActionCompletion time
```

在设置的 time 超时的情况下，任何工作线程都被关闭。在关机时要保存数据，可以使用：

```
$objectQueueTimeoutSaveOnShutdown time
```

如歌按上面那样设置了，所有队列的数据在 rsyslogd 终止前都会被存储到磁盘。

6.3.4.3. 使用 rsyslog 队列新语法

rsyslog 7 提供了新的语法，队列对象在 action（）内部定义，既可以单独使

用或在/etc/rsyslog.conf 中一个规则集使用。执行队列的格式如下：

```
action(type="action_type" queue.size="queue_size"
queue.type="queue_type" queue.filename="file_name")
```

用要执行的操作的模块名称替换 action_type，用消息队列可以容纳的最大数目替代的 queue_size。对于 queue_type，可以选择 disk 队列或从 in-memory 队列中选择一个：direct，LinkedList 的或 fixedarray。对于 file_name 仅指定一个文件名，而不是路径。请注意，如果创建一个新的目录来保存日志文件，SELinux 必须被设置。

6.3.5. 在日志服务器上配置 rsyslog

rsyslog 服务提供的设施可用用来运行日志服务器和配置单个系统来发送日志文件到日志服务器。请参见实例“可靠的转发日志消息到服务器”查看客户端 rsyslog 配置信息。

rsyslog 服务必须安装在您打算作为日志服务器的系统上，并且将所有的系统配置为将日志发送给日志服务器。中标麒麟高级服务器操作系统软件 V7.0 默认情况下会安装 rsyslog，如果需要，以确保系统确实安装了 rsyslog，以 root 身份输入以下命令：

```
~]# yum install rsyslog
```

Syslog 流量默认的协议和端口是 UDP 和 514，其在/ etc/services 文件中列出。然而，rsyslog 默认使用 TCP 在端口 514。在配置文件/etc/rsyslog.conf 中，TCP 是由@@声明。

其它的端口在示例有时会用到，然而 SELinux 的默认配置为仅允许发送和接收在下面列出的端口中：

```
~]# semanage port -l | grep syslog
```

semanage 实用程序由 policycoreutils-Python 包的一部分提供。如果需要的话，安装包如下：

```
~]# yum install policycoreutils-python
```

此外，在默认情况下 SELinux 的 rsyslog 类型是 rsyslogd_t，其被配置为允许发送和接收类型是 rsh_port_t 的远程 shell（RSH）端口，其 TCP 默认端口是 514。因此，这是没有必要使用 semanage 的明确的允许 TCP 端口为 514。例如，要检查 SELinux 允许什么程序在端口 514 通信，输入命令如下：

```
~]# semanag eport -l | grep 514
```

请在您打算做日志服务器的系统上，使用下列过程中的步骤。在这些过程中的所有步骤必须以 root 身份户执行命令。

6.3.5.1. 在日志服务器上使用模板

rsyslog7 具有多种不同的模板样式。字符串模板最接近旧格式。用字符串格式生成上面例子中的模板，如下所示：

```
template(name="TmplAuthpriv" type="string"
string="/var/log/remote/auth/%HOSTNAME%/%PROGRAMNAME:::secpathr
eplace%.
log"
)
template(name="TmplMsg" type="string"
string="/var/log/remote/msg/%HOSTNAME%/%PROGRAMNAME:::secpathr
eplace%.
log"
)
```

这些模板也可以写成下面这样的格式列表：

```
template(name="TmplAuthpriv" type="list") {
constant(value="/var/log/remote/auth/")
property(name="hostname")
constant(value="/")
property(name="programname" SecurePath="replace")
constant(value=".log")
}
template(name="TmplMsg" type="list") {
constant(value="/var/log/remote/msg/")
property(name="hostname")
constant(value="/")
property(name="programname" SecurePath="replace")
constant(value=".log")
}
```

对这些 rsyslog 新规则来说，这个模板文本格式可能会更容易阅读，因此可以更容易应用。

要完成新语法的更改，我们需要重新生成模块加载命令，添加规则集，然后绑定协议，端口和规则集的规则：

```
module(load="imtcp")
ruleset(name="remote1"){
authpriv.*  action(type="omfile"  DynaFile="TmplAuthpriv")
*.info;mail.none;authpriv.none;cron.none  action(type="omfile"
DynaFile="TmplMsg")
}
input(type="imtcp"  port="10514"  ruleset="remote1")
```

6.3.6. 使用 Rsyslog 模块

由于采用模块化设计，rsyslog 提供各种模块，这些模块提供额外的功能。需要注意的是模块可以由第三方开发。大多数模块提供额外的输入（见下文输入模块）或输出（见下文输出模块）。其他模块提供具体到每个模块的特定功能。加载一个模块后，其可提供可用的附加配置指令。要加载模块，请使用以下语法：

```
$ModLoad  MODULE
```

其中$ModLoad 是加载指定的模块的全局指令。MODULE 表示您所需的模块。例如，如果您要加载的文本文件输入模块（imfile），使 rsyslog 将任何标准的文本文件转换成系统日志信息，在 /etc/rsyslog.conf 配置文件中指定以下行：

```
$ModLoad  imfile
```

rsyslog 提供了许多模块，这些模块被分为以下类别：

- 输入模块-输入模块收集来自各种来源的消息。输入模块的名称总是以 im 前缀开始，如 imfile 和 imjournal。
- 输出模块-输出模块提供便利给各种目标发出消息，如通过网络发送，存储在数据库中，或者加密等消息。输出模块的名称总是以 om 前缀，如 om snmp，omrel p 等等。
- 解析模块- 这些模块在创建自定义解析规则，或者解析异常的消息是有用的。使用 C 语言的中间层，您可以创建自己的消息解析器。解析器模块的名称总是以 pm 前缀开始，如 pmrfc5424，pmrfc3164，等。
- 消息修改模块 - 消息修改模块修改系统日志消息的内容。这些模块的名称以 mm 的前缀开始。消息修改的模块如 mmanon，mmnormalize，或 m mjsonparse 其用于匿名或消息的正常化。
- 字符串生成模块 - 串生成模块生成基于消息内容的字符串，并与 rs

yslog 提供的模板功能相互协同。有关模板的详细信息，请参阅 5.3.2.3 模板。串生成模块的名称总是以 sm 前缀开始，如 smfile 或 smtrad 文件。

- 库模块 - 库模块为其他可加载模块提供功能。这些模块在 rsyslog 需要时自动被加载，不能由用户配置。

所有可用的模块和它们的详细描述的完整列表可以在 http://www.rsyslog.com/doc/rsyslog_conf_modules.html 找到。

警告：

需要注意的是 rsyslog 加载的模块，这使他们可以访问自己的函数和数据。这会带来潜在的安全威胁。为了最大限度地减少安全风险，仅使用可信赖模块。

6.3.6.1. 导入 text 文件

文本文件输入模块，简称 imfile，使 rsyslog 将任意文本文件转化成系统日志消息流。您可以使用 imfile 从创建自己的文本文件日志的应用程序中导入日志消息。要加载 imfile，在/etc/rsyslog.conf 添加以下内容：

```
$ModLoad imfile
$InputFilePollInterval int
```

它足以加载 imfile 一次，导入多个文件时也是如此。$InputFilePollInterval 全局指令指定 rsyslog 检查被连接的文本文件变化的频率。默认的时间间隔为 10 秒，要改变它，用以秒为单位的时间间隔替换 int。

为了确定文本文件导入，在 /etc/rsyslog.conf 文件中使用的语法如下：

```
# File 1
$InputFileName path_to_file
$InputFileTag tag:
$InputFileStateFile state_file_name
$InputFileSeverity severity
$InputFileFacility facility
$InputRunFileMonitor
# File 2
$InputFileName path_to_file2
...
```

需要 4 个设置来指定一个输入文件：

- 用文本文件路径替代 path_to_file
- 用消息的 tag 名替代 tag
- 用唯一的状态文件名替代 state_file_name。状态文件都存储在 rsyslo

g 的工作目录，为监控的文件保持标记，标记已经被处理完的分区。如果删除它们，整个文件将被重新读入。请确保您指定了一个不存在的名称。

- 添加$InputRunFileMonitor 指令使能文件监控。没有这个设置，文本文件会被忽略。

除了所要求的指令，存在可以应用到文本输入的几个其它设置。通过用适当的关键字替换关键字 severity 来设置导入消息的严重性。用关键字替换 facility 来定义生成消息的子系统。

6.3.6.2. 从数据库导出消息

在数据库中，处理日志数据比处理文本文件可以更快，更方便。基于所使用的 DBMS 的类型，选择各种输出模块，例如 ommysql，ompgsql，omoracle，或 ommongodb。作为替代，使用依赖于 libdbi 库的通用 omlibdbi 输出模块。omlibdbi 模块支持的数据库系统有 Firebird/ Interbase，MS SQL，Sybase，SQLite，ingres，甲骨文，mSQL，MySQL 和 PostgreSQL。

6.3.6.3. 使能加密传输

网络传输的机密性和完整性可以通过 TLS 或 GSSAPI 加密协议来提供。

传输层安全性（TLS）是旨在提供通过网络通信的安全性的加密协议。当使用 TLS，rsyslog 消息被发送之前加密，发送者和接收者之间的需要相互认证。

通用安全服务 API（GSSAPI）是一种为程序访问安全服务的应用程序编程接口。为了在与 rsyslog 的连接中使用，您必须有一个正常的 Kerberos 环境。

6.3.7. Syslogd 服务和日志的交互

如上所述，rsyslog 和 journal，您系统上的两个日志应用程序，有几个鲜明的特点，使它们适用于特定的用例。在很多情况下（见 6.4Syslogd 日志结构），它们可以相互合作，比如创建结构化的信息，并将它们存储在一个文件数据库中。需要这种合作的一个通信接口是由输入和输出模块提供的，这些模块由 Rsyslog 和 journal 的通信 socket 支持。

默认情况下，rsyslogd 使用 imjournal 模块作为日志文件的默认的输入模式。使用这个模块，您不仅可以导入消息，也可以导入由 Journald 提供的结构化数据。此外，旧的数据可以从 journald 导入（除非禁止用$ ImjournalIgnorePreviousMessages 指令）。参考 6.4.1 从日志中导入数据部分关于 imjournal 的基本配置。

作为替代，配置 rsyslogd 来读取由 journal 提供的套接字作为一个基于 syslog 应用的输出。套接字的路径/run/systemd/journal/ syslog。当您想维护 rsyslog 消息，请使用此选项。相比 imjournal，套接字输入提供更多的功能，如绑定规则集或过滤器。要通过套接字导入日志数据，在/etc/rsyslog.conf 中使用以下配置：

```
$ModLoad imuxsock
$OmitLocalLogging off
```

上述语法加载 imuxsock 模块并关闭$OmitLocalLogging 指令，使能通过系统套接字导入。套接字的路径分别在/etc/rsyslog.d/listen.conf 中指定，如下：

```
$SystemLogSocketName /run/systemd/journal/syslog
```

您也可以从 Rsyslog 输出消息到使用 omjournal 模块的 journal。在/etc/rsyslog.conf 中配置输出如下：

```
$ModLoad omjournal
*.* :omjournal:
```

例如，下面的配置转发所有收到的信息通过 TCP 端口 10514 到 journal：

```
$ModLoad imtcp
$ModLoad omjournal
$RuleSet remote
*.* :omjournal:
$InputTCPServerBindRuleset remote
$InputTCPServerRun 10514
```

6.4. Syslogd 日志结构

在产生大量的日志数据的系统上，一个结构化格式的日志信息可以方便地被维护。通过结构化的信息，很容易搜索特定信息，生成统计信息和处理消息的改变和不一致。rsyslog 使用 JSON（JavaScript 对象符号）格式提供日志信息的结构化。

比较下面非结构化的日志消息：

```
Oct 25 10:20:37 localhost anacron[1395]: Jobs will be executed
sequentially
```

下面是结构化的日志消息：

```
{"timestamp":"2020-2-7 T10:20:37", "host":"localhost",
"program":"anacron", "pid":"1395", "msg":"Jobs will be executed
sequentially"}
```

使用键值对搜索结构化数据比使用正则表达式搜索文本文件的速度更快更精确。结构话消息还可以让您搜索各种应用程序生成的相同的消息。另外，JSON 文件可以存储在文档数据库，如 MongoDB，它提供了额外的性能和分析功能。另一方面，一个结构化的消息需要比非结构化的消息更多的磁盘空间。

在 rsyslog，带有元数据的日志信息被使用 imjournal 模块的 journal 推出。使用 mmjsonparse 模块，可以解析来自 journal 和其他来源导入的数据，并进一步对其进行处理，例如，作为一个数据库输出。为了使解析成功，mmjsonparse 要求输入消息是结构化的，向其在 lumberjack 项目定义的那样。

Lmberjack 项目的目的是以向后兼容的方式为 rsyslog 增加结构化日志记录。要确定一个结构化的消息，Lmberjack 指定@ cee：字符串，其前置 JSON 结构。另外，Lmberjack 定义了应该用在 JSON 串标准字段名称的列表。有关 Lmberjack 的更多信息，请参见在线文档。

下面是 lumberjack 格式消息的例子：

```
@ cee: {"pid":17055, "uid":1000, "gid":1000, "appname":"logger",
"msg":"Message text."}
```

要在 Rsyslog 中构建这个结构，一个模板会被使用，请参见 6.4.2 过滤结构化消息。应用程序和服务器可以采用 libumberlog 库来生成 lumberjack 兼容形式的消息。有关 libumberlog 的更多信息，请参见在线文档 。

6.4.1. 从日志中导入数据

imjournal 模块是 rsyslog 的输入模块用来读取日志文件（见 6.3.7SSyslogd 服务和日志的交互。作为其它的 rsyslog 消息，日志消息以文本格式被记录。然而，随着进一步的处理，其能够将由 journal 提供的元数据翻译成结构化的消息。

为了从 Journal 导入数据到 Rsyslog，在/etc/rsyslog.confi 文件中使用下面配置：

```
$ModLoad imjournal
$imjournalPersistStateInterval number_of_messages
$imjournalStateFile path
$imjournalRatelimitInterval seconds
$imjournalRatelimitBurst burst_number
$ImjournalIgnorePreviousMessages off/on
```

- 使用 number_of_messages，您可以指定保存日志数据的频率。每当达到 number_of_messages 指定的消息数时，就会触发数据保存。
- 用状态文件的路径替代 path。此文件跟踪 journal 记录，其是处理的最后一个记录。
- 使用 seconds，设置的限制速率的时间间隔。此间隔期间处理的消息的数量不能超过在 burst_number 指定的值。默认设置为每 600 秒 20000 的消息。Rsyslog 会丢弃在规定的时限内超过最大 burst_number 值的消息。
- 使用$ ImjournalIgnorePreviousMessages 可以忽略当前在 journal 的消

息和只导入新的消息，这些消息当未指定状态文件时被使用。默认设置为关闭。请注意，如果该设置是关闭的，没有状态文件，所有 journal 的消息会被处理，即使他们已经在 rsyslog 以前的会话中被处理过。

注意：

您可以同时使用 imjournal 和 imuxsock 模块，imuxsock 模块是旧的系统日志输入。然而，为了避免消息重复，您必须阻止 imuxsock 读取 journal 的系统套接字。要做到这一点，使用$OmitLocalLogging 指令：

```
$ModLoad imuxsock
$ModLoad imjournal
$OmitLocalLogging on
$AddUnixListenSocket /run/systemd/journal/syslog
```

您可以通过将存储在 journal 的数据和元数据翻译成结构化消息。其中的一些元数据项在本章实例“journalctl 的 verbose 输出”中列出，想获取 journal 域的完整列表，请参阅 systemd.journal-fields（7）手册页。例如，可以将重点放在内核 jouranl 字段，该域被内核初始生成的消息使用。

6.4.2. 过滤结构化消息

要创建一个 rsyslog 解析模块需要的 lumberjack 格式的消息，请使用以下模板：

```
template(name="CEETemplate" type="string" string="%TIMESTAMP% %HOSTNAME%
%syslogtag% @ cee: %$!all-json%\n")
```

这个模板预先考虑@cee：可以应用的 JSON 字符串，例如，当创建使用 om file 模块的输出文件时。要访问 JSON 字段名称，使用$！前缀。例如，搜索符合下面的筛选条件的消息，指定了消息的主机名和 UID。

```
($!hostname == "hostname" & & $!UID== "UID")
```

6.4.3. 解析 JSON

mmjsonparse 模块用于解析结构化消息。这些信息可以来自 journal 或其他输入源，并且必须由 lumberjack 项目中定义的方式进行格式化。这些消息是由@cee：字符串表示而被识别的。然后，mmjsonparse 会检查 JSON 结构是否是有效的，然后该消息被解析。

为了用 mmjsonparse 模块解析 lumberjack 格式的 JSON 消息，在/etc/rsyslog.conf 文件下使用下面的配置：

```
$ModLoad mmjsonparse
*.* :mmjsonparse:
```

在这个例子中，mmjsonparse 模块在第一行被加载，所有的消息转发给它。目前，没有可用的配置参数用于 mmjsonparse。

6.4.4. 向 MongoDB 中存储消息

rsyslog 支持通过 ommongodb 输出模块在 MongoDB 的文档型数据库中存储 JSON 日志。

要转发日志消息到 MongoDB 中，在/etc/rsyslog.conf 使用以下语法（ommongodb 配置参数只适用于新配置格式;见 6.3.3 使用新的配置格式）

```
$ModLoad ommongodb
*.* action(type="ommongodb" server="DB_server" serverport="port"
db="DB_name" collection="collection_name" uid="UID" pwd="password
")
```

- 用 MongoDB 服务器的地址或者名字替代 DB_server. 配置端口需要从 MongoDB 的服务器选择一个非标准的端口。默认端口值是 0，并且通常没有必要改变该参数。
- 使用 DB_NAME，确定您想直接输出到 MongoDB 的服务器上的哪个数据库。用这个数据库集合的名称替换 collection_name。在 MongoDB 中，集合是一组文档，它和一个 RDBMS 表是等效的。
- 您通过设置 UID 和 password 来输入您的细节信息。

您可以使用模板来塑造最终数据库的输出形式。默认情况下，rsyslog 使用的模板基于标准的 lumberjack 字段名称。

6.5. 调试 Rsyslog

在 debug 模式运行 rsyslogd,使用如下命令：

```
rsyslogd -dn
```

使用此命令，rsyslogd 产生调试信息并打印到标准输出。-n 表示“no fork”。您可以修改调试环境变量，例如，可以在日志文件中存储调试输出信息。在启动 rsyslogd 之前，在命令行上执行以下操作：

```
export RSYSLOG_DEBUGLOG="path"
export RSYSLOG_DEBUG="Debug"
```

用所需记录调试信息文件的位置替换 path。对于可用于 RSYSLOG_DEBUG 变量选项的完整列表，请参阅 rsyslogd（8）手册的相关章节。

检查在/etc/rsyslog.conf 文件中使用的语法是否有效：

```
rsyslogd -N 1
```

其中，1 表示输出消息的详细程度的级别。这是前向兼容性选项，因为目前，只有一个级被提供。但是，您必须添加此参数来运行验证。

6.6. 使用日志

journal 是 systemd 的一个组件，它负责查看和管理日志文件。它可并行使用或者代替一个旧的 syslog 守护进程，如 rsyslogd。journal 的开发是为了解决与旧的日志记录有关的问题。它紧密地与系统的其余部分集成，支持多种日志记录技术和访问管理日志文件。

记录数据由 journald 服务收集，存储，并处理。它创建和维护名为 journals 的二进制文件，这些记录的信息来自内核，用户进程，标准输出，标准错误输出，或通过原生 API 的标准错误输出。这些 journals 被结构化并被索引，它提供了比较快的查询道时间。journal 条目有一个唯一的标识符。journald 服务收集每个日志消息众多的元数据字段。日志文件是安全的，不能被手动编辑。

6.6.1. 查看日志文件

为了访问 journal 日志，使用 journalctl 工具。以 root 身份查看日志类型的基本信息：

```
journalctl
```

此命令的输出是系统所有日志文件的列表，包括系统组件和用户产生的消息。该输出的结构类似于/var/log /messages 文件中消息结构， 但有一定的改进：

- 记录优先级的标记是可见的。错误的优先级和更高的优先级使用了红色，通知和警告优先线使用加粗的字体。
- 时间戳转换为系统的本地时区。
- 所有记录数据都会显示，包括轮替的日志。
- 引导的开始将被标上专用线。

6.6.2. 访问控制

默认情况下，没有 root 权限的 journal 用户，只能查看由它们产生的日志文件。系统管理员可以添加选定用户到 adm 组，授予他们访问完整日志文件的权限。要做到这一点，以 root 用户输入如下：

```
usermod -a -G adm username
```

用要添加到 adm 组的用户名替代 username。该用户随后接收 journalctl 命令的输出同 root 用户获取的输出一样。请注意，访问控制只有当为 journal 启用了持久存储才会工作。

6.6.3. 使用 Live view

当不带参数调用，journalctl 从收集到的最早的条目开始，显示条目的完整列表。有了 live view，您可以监控实时的日志消息，因为新的条目出现就会被打印出来。要开启 journalctl 的 live view 模式，输入：

```
journalctl -f
```

该命令返回十个最新的日志行列表。journalctl 工具将保持运行，等待新的消息并立即显示他们。

6.6.4. 过滤消息

不带参数执行 journalctl 命令的输出信息是很多的，因此您可以使用各种过滤方法来提取信息，以满足您的需求。

通过优先级过滤

日志消息通常用于跟踪系统上错误的行为。要查看选定的或更高优先级消息，请使用以下语法：

```
journalctl -p priority
```

在这里，用下面的关键字之一（或数字）替换优先级：debug（7），info（6），notice（5），warning（4），err（3），crit（2），alert（1），和 emerg（0）。

实例：通过优先级过滤

查看优先级为 error 或更高的消息，使用：

```
journalctl -p err
```

通过时间过滤

要查看仅从当前引导的日志条目：

```
journalctl -b
```

如果您偶尔会重新启动系统，-b 不会显著减少 journalctl 的输出。在这样的情况下，基于时间的滤波是更有帮助：

```
journalctl --since=value --until=value
```

使用--since 和--until 选项。您可以查看指定时间范围内创建的日志信息。您

可以以 time 的格式或者 data 的格式或者两者都可传递 value 给这两个选项，如以下示例。

实例：通过优先级和时间过滤

根据具体的要求过滤选项可以组合，以减少的结果输出。例如，从某个时间点，查看警告或更高优先级的消息，如下：

```
# journalctl -p warning --since="2020-3-5 23:59:59"
```

高级过滤

在本章实例“journalctl 的 verbose 输出”列出指定日志条目的一组字段，其都可以用于过滤。对于 systemd 可存储的元数据的完整说明，参见 systemd.jounal-fields（7）手册页。每个日志信息的元数据都被收集，而无需用户干预。值通常是基于文本的，但可以采取二进制和大的值; 虽然不是很常见的，但是字段可以分配多个值。

要查看发生在一个特定领域产生的唯一值的列表，请使用以下语法：

```
journalctl -F fieldname
```

用您想要的字段名替代 fieldname。

要显示出满足特定条件日志条目，请使用以下语法：

```
journalctl fieldname=value
```

用字段名和该字段包含的值替代 fieldname 和 value。结果，符合这个条件的行会输出。

注意：

如通过 systemd 存储的元数据字段的数量是相当大的，它很容易忘记感兴趣的字段的确切名称。如果不能确定，请键入：

```
journalctl
```

按两次 tab 键。这会显示出可用的字段名。Tab 键名称补齐是基于该字段的上下文选项完成的，您可以输入一部分字母，然后按 Tab 键来自动完成这个名字。同样，您可以从一个字段中列出唯一值。 如下：

```
journalctl fieldname=
```

按 tab 键两次，就会像输入 journalctl –F fieldname 一样。

您可以在一个字段自定多个值，如下：

```
journalctl fieldname=value1 fieldname=value2 ...
```

同一个字段指定两个可以匹配的值，像逻辑 or 一样匹配。符合匹配值 1 或值 2 的条目会显示。

当然，您可以指定多个 field-value 对来进一步减少输出：

```
journalctl fieldname1=value fieldname2=value ...
```

如果指定了两个不同的字段名的匹配，它们将与逻辑 AND 组合。必须匹配这两个条件的条目才会显示。

使用+号，您可以设置逻辑 or 组合匹配多字段：

```
journalctl fieldname1=value + fieldname2=value ...
```

该命令返回至少匹配一个条件的条目，不仅是满足所有条件的条目。

实例：高级过滤器设置

要显示通过 avahi-daemon 或 crond.service 创建的条目。且该条目的用户 UID 为 70，请使用以下命令：

```
journalctl _UID=70 _SYSTEMD_UNIT=avahi-daemon.service _SYSTEMD_UNIT=crond.service
```

由于_SYSTEMD_UNIT 字段可以设置两个值，匹配两者的结果将显示，但_UID= 70 条件必须被满足。这可以简单地表示为（UID=70 and (avahi or cron)）.

您可以应用到前面提到的过滤器，其以 live-view 模式来跟踪选定组的日志消息的变化：

```
journalctl -f fieldname=value ...
```

6.6.5. 使能持续存储

默认情况下，journal 只在内存中或/run/log/journal/目录下的小环形缓冲区中存储日志文件。使用 journalctl 显示最近历史日志就足够了。该目录有易失性，日志数据不会永久保存。在默认配置下，syslog 读取日志记录，并将其存储在/ var/log/目录下。若持续性日志记录开启，日志文件存储在/ var/log/journal，这意味着重新启动后他们仍然存在。对一些用户，journal 能代替 rsyslog。

使能持续性存储有下面的优点：

- 较长时间的可用来解决问题的更丰富的数据会被记录。
- 对于需要立刻解决的问题，重启后可以获取更丰富的可用的数据。
- 服务器控制台当前从日志中读取数据，而不是日志文件。

持续性存储也有某些缺点

- 持续性存储所存储的数据量取决于空闲的内存，不保证可以覆盖特定的时间跨度。
- 需要更多的磁盘空间存储日志文件。

journal 启用持续性存储，按下面所示的例子手动创建 journal 目录，以 root 用户输入：

```
mkdir -p /var/log/journal
```

重启 journald 来是设置生效。

```
systemctl restart systemd-journald
```

6.7. 在图形界面管理日志

作为一种替代上述命令行的实用程序，中标麒麟高级服务器操作系统软件 V7.0 提供了一个可访问的图形用户界面来管理日志消息。

6.7.1. 查看日志文件

大多数日志文件都以纯文本格式存储。您可以用任何文本编辑器如 vi 或 Emacs 来查看它们。某些日志文件可以被系统上所有用户查看;然而，需要 root 权限来阅读大多数日志文件。

以交互的实时的方式查看系统日志文件的应用程序。可以使用 SystemLog。

注意：

为了可以使用 SystemLog，首先确保您的系统安装了 gnome-system-log 包。以 root 用户执行如下：

```
~]# yum install gnome-system-log
```

更多关于使用 yum 安装包的信息，请查看 2.2.4 安装软件包。

在您安装完 gnome-system-log 包之后，通过点击应用程序->系统工具->SystemLog 或者输入如下命令：

```
~]$ gnome-system-log
```

应用程序只会显示存在的日志文件，因此列表也许和图 6-5 SystemLog 显示的有所不同。

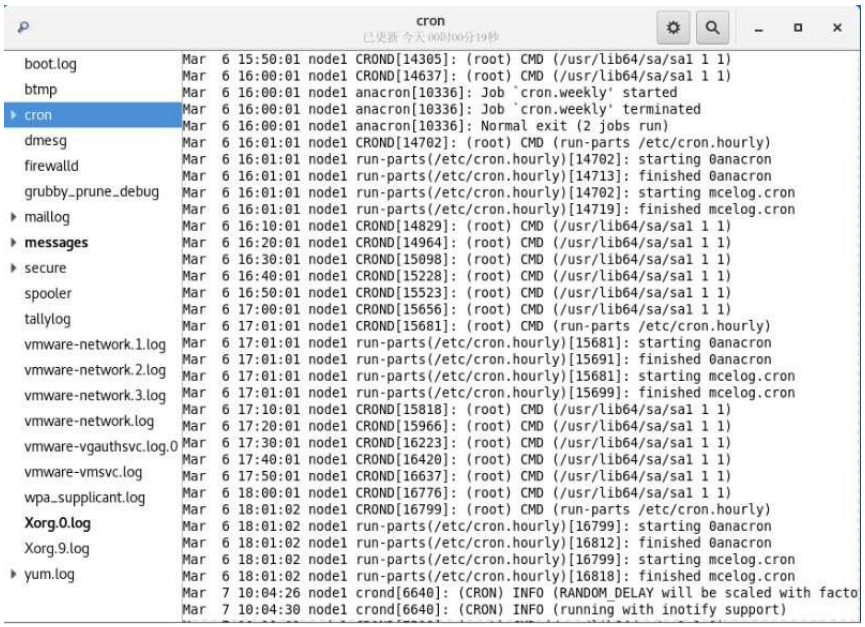


图 6-5 SystemLog

SystemLog 应用程序允许您过滤任何存在的日志文件。点击标有齿轮符号的按钮来查看菜单，选择“齿轮（首选项）->文件->管理过滤器”来定义或编辑所需的过滤器。

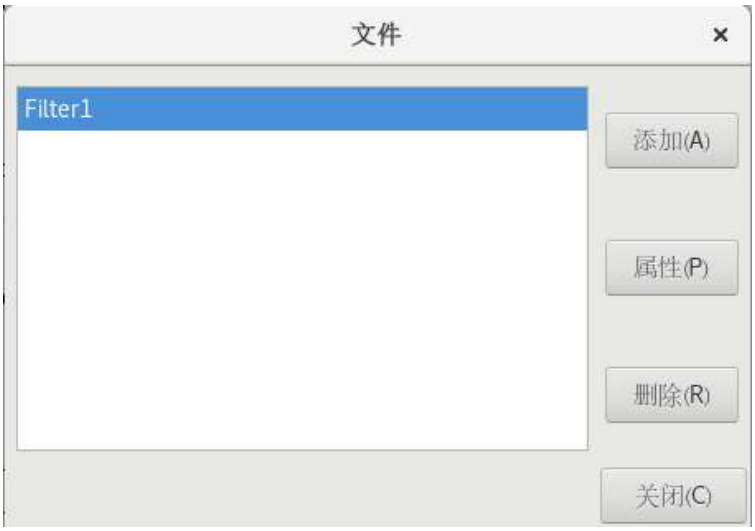


图 6-6 系统日志过滤器

图 6-7 添加新的过滤器

添加或编辑过滤器，您可以定义它的参数，该参数在图图 6-7 添加新的过滤器中显示。

当定义一个过滤器，下面的参数需要被编辑。

- 名称 – 定义过滤器的名字
- 正则表达式 -指定被应用到日志文件的正则表达式，其会尝试匹配文本中任何可能的字符串。
- 效果
  - 高亮 - 如果选中，找到的结果将用所选择的颜色突出显示。 可以选择是否要突出背景或文本的前景
  - 隐藏 - 如果选中，发现的结果会从您正在查看的日志文件中被隐藏。

当您定义了至少一个过滤器，它可以在 Filters 菜单中选择，它会自动搜索您在过滤器中定义的字符串，并突出显示或隐藏当前正在查看日志文件中每一个成功匹配的消息。

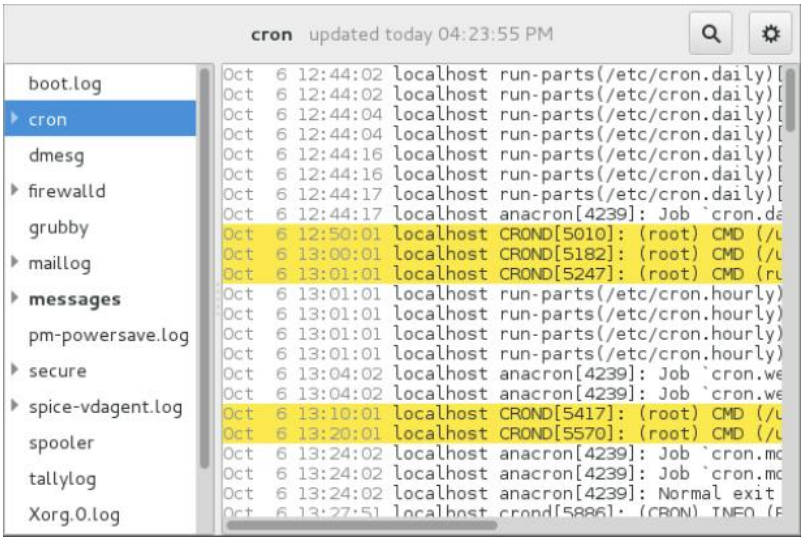


图 6-8 SystemLog – 启用过滤器

当您选择“仅显示匹配项”选项，只有匹配的字符串将显示在当前正在查看日志文件中。

6.7.2. 添加日志文件

要添加一个日志文件到您想查看的列表中，选择“齿轮->打开”，这会显示“打开日志”窗口，在这个窗口您可以选择您想查看的目录和日志文件。图 6-9 System Log – 添加日志文件显示了“打开日志”窗口。

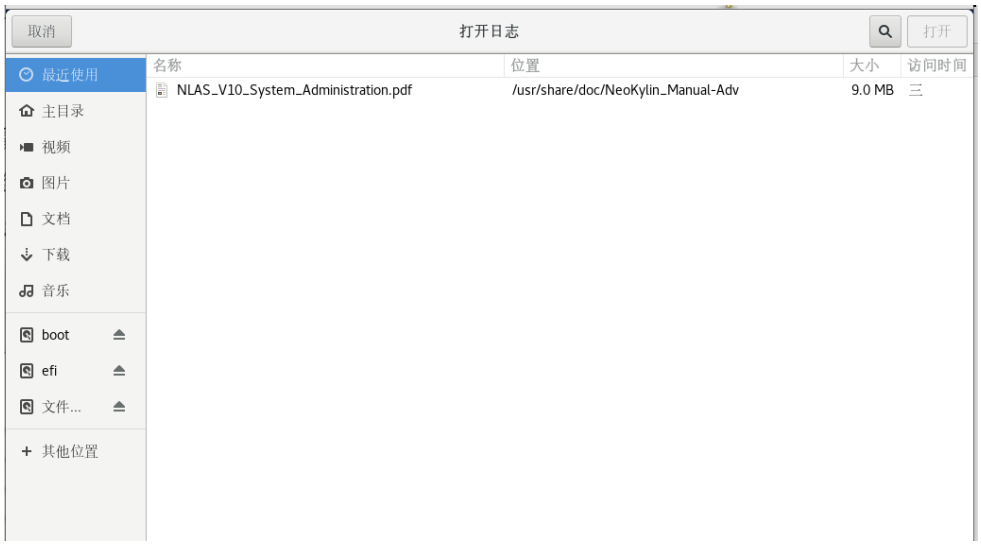


图 6-9 System Log – 添加日志文件

点击 Open 按钮来打开文件，文件会立即添加到您正查看的列表，这里您可以选中它查看他的内容。

注意：System Log 也允许您打开以.gz 格式压缩的日志文件。

6.7.3. 监控日志文件

System Log 以在默认情况下监控所有打开的日志。如果一个新行被添加到受监视的日志文件，日志名称以粗体显示在日志列表中。如果日志文件被选中或显示，新的行以粗体显示在日志文件的底部。图 6-10 System Log - 新日志警报说明了一个新的警报在 cron 日志文件中和在 messages 日志文件中。点击 messages 日志文件,日志在文件中以新行粗体显示。

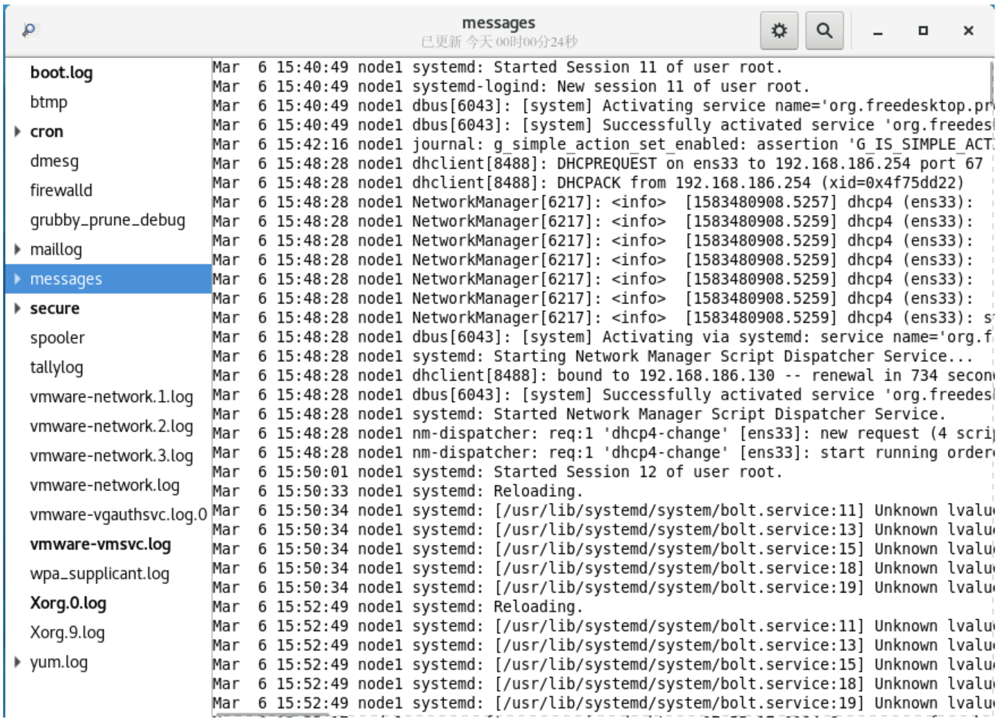


图 6-10 System Log - 新日志警报

6.8. 自动化系统任务

任务，也被称为工作，可以通过配置，在一个指定的日期，在一个指定的时间周期内，或者当系统的平均负载低于 0.8 时，自动地运行。

中标麒麟高级服务器操作系统软件预先配置了运行一些重要的系统任务，以保持系统的更新。例如，locate 命令使用的 slocate 数据库每天都会更新。系统管理员可以使用自动任务来执行周期性的备份、监控系统、运行自定义脚本等等。

中标麒麟高级服务器操作系统软件提供了以下自动任务工具：cron、anacron、at 和 batch。

每一种工具都是为了调度不同的任务类型而被设计的，Cron 和 Anacron 调度重复发生的任务，At 和 Batch 调度一次性的任务（请分别参考 6.8.1Cron 和 Anacron 和 6.8.8At 和 Batch）。

中标麒麟高级服务器操作系统软件 V7.0 支持使用 systemd.timer 在一个指定

的时间执行一个任务。参见 systemd.timer(5)用户手册页面了解更多信息。

6.8.1. Cron 和 Anacron

Cron 和 Anacron 都是能够调度重复性任务在一个确定的时间点执行的守护进程，时间点是通过准确的时间、月份中的某天、月份、一周中的某天、周来定义的。

Cron 的任务可以每分钟都运行。但是，这个工具假定系统是持续运行的，如果在任务被安排的时间系统并没有运行，则任务不会被执行。

另一方面，如果系统在任务被安排的时间被没有运行，Anacron 会记住这个已经被安排的任务。一旦系统开始运行，这个任务将被马上执行。然而，Anacron 对于一个任务一天只能运行一次。

6.8.2. 安装 Cron 和 Anacron

要安装 Cron 和 Anacron，您需要安装 Cron 的 cronie 包和 Anacron 的 cronie-anacron 包（cronie-anacron 是 cronie 的一个子包）。

要确定您的系统上是否已经安装了相应的包，可以执行以下命令：

```
rpm -q cronie cronie-anacron
```

如果已经安装了，上述命令将返回 cronie 包和 cronie-anacron 包的完整名称，否则将通知您相应的包不可用。

要安装这些包，以 root 用户按照下面的格式来使用 yum 命令：

```
yum install package
```

例如，要同时安装 Cron 和 Anacron，可以在命令行提示符下输入以下命令：

```
~]# yum install cronie cronie-anacron
```

要了解如何在中标麒麟高级服务器操作系统软件中安装新的包，请参见 2.2.4 安装软件包。

6.8.3. 运行 Crond 服务

cron 和 anacron 任务都是由 crond 服务来控制的。本节将说明如何启动、停止和重启 crond 服务，以及如何配置让其开机自启动。要了解更多有关在中标麒麟高级服务器操作系统软件 V7.0 中如何管理服务的通用方法，请参见 3.1 使用 systemd 管理系统服务。

6.8.3.1. 启动 Cron 服务

要确定服务是否正在运行，请使用以下命令：

```
systemctl status crond.service
```

要在当前会话中运行 crond 服务，请以 root 用户在命令行提示符下输入以下命令：

```
systemctl start crond.service
```

要配置服务开机自启动，请以 root 用户执行以下命令：

```
systemctl enable crond.service
```

6.8.3.2. 停止 Cron 服务

要在当前会话中停止 crond 服务，请以 root 用户在命令行提示符下输入以下命令：

```
systemctl stop crond.service
```

要禁止服务开机自启动，请以 root 用户执行以下命令：

```
systemctl disable crond.service
```

6.8.3.3. 重启 Cron 服务

要重启 crond 服务，请以 root 用户在命令行提示符下输入以下命令：

```
systemctl restart crond.service
```

该命令停止服务后将很快地再次启动服务。

6.8.4. 配置 Anacron 任务

调度任务的主要配置文件是/etc/anacrontab 文件，它只能通过 root 用户进行访问。该文件包含以下内容：

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22
#period in days   delay in minutes   job-identifier   command
1       5       cron.daily              nice run-parts /etc/cron.daily
7       25      cron.weekly             nice run-parts /etc/cron.weekly
```

```
@monthly 45 cron.monthly nice run-parts /etc/cron.monthly
```

前三行定义用来配置 anacron 任务运行环境的相关变量：

- SHELL——用来运行任务的 shell 环境（示例中是 Bash shell）
- PATH——可执行程序的路径
- MAILTO——通过电子邮件接收 anacron 任务输出的用户的名称

如果没有定义 MAILTO 变量（MAILTO=），则不会发送邮件。

接下来的两个变量用来修改已定义任务的调度时间：

- RANDOM_DELAY——将会加到为每个任务指定的 delay in minutes 变量上的最大分钟数。

默认情况下，最小延迟的值被设置为 6 分钟。

例如，如果 RANDOM_DELAY 被设置为 12，那么这个特定的 anacrontab 中的每一个任务的 delay in minutes 值都将被加上 6 到 12 分钟。RANDOM_DELAY 的值也可以设置为 6 以下，包括 0。当设置为 0 的时候，则不会增加随机延迟。随机延迟被证明是很有用的，例如，当多台共享一个网络连接的计算机需要每天都下载相同的数据时。

- START_HOURS_RANGE——计划任务可以运行的时间段，以小时为单位。

万一错过了这个时间段，例如，由于停电等原因，则当天的计划任务不会被执行。

/etc/anacrontab 文件的剩余行代表计划任务，并且遵从以下格式：

```
period in days delay in minutes job-identifier command
```

- period in days——按天数计的任务执行频率

该属性值可以被定义为一个整数或者一个宏（@daily、@weekly 或者@monthly），@daily 和整数 1 表示的是同一个值，@weekly 和整数 7 一样，而@monthly 则表示任务一个月只运行一次，不管这个月的天数是多少。

- delay in minutes——在执行任务前，anacron 等待的分钟数

该属性值可以被定义为一个整数。如果该值被设置为 0，则表示没有延迟。

- job-identifier——用于日志文件中关联一个特定任务的唯一名称
- command——将被执行的命令

这里的命令既可以是一个类似 ls /proc >> /tmp/proc 的命令，也可以是一个执行自定义脚本的命令。

以井号（#）开头的所有行都是注释，不会被处理。

6.8.4.1. Anacron 任务示例

以下是一个简单的/etc/anacrontab 文件的示例：

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=30
# the jobs will be started during the following hours only
START_HOURS_RANGE=16-20

#period in days   delay in minutes   job-identifier   command
1         20          dailyjob                nice run-parts /etc/cron.d
aily
7         25          weeklyjob               /etc/weeklyjob.bash
@monthly  45          monthlyjob              ls /proc >> /tmp/proc
```

在这个 anacrontab 文件中定义的所有任务，都将随机的延迟 6 到 30 分钟，并且将在 16:00 到 20:00 之间被运行。

第一个被定义的任务将在每天 16:26 到 16:50 间被触发（RANDOM_DELAY 是在 6 到 30 分钟之间；delay in minutes 属性值为 20 分钟）。该任务所指定的命令将使用 run-parts 脚本（run-parts 脚本接受一个目录作为命令行参数，并顺序地执行目录中的每一个程序）执行/etc/cron.daily/目录下的所有程序。参见 run-parts 用户手册页面了解更多有关 run-parts 脚本的信息。

第二个任务每周执行一次/etc/目录下的 weeklyjob.bash 脚本。

第三个任务运行一个命令，该命令把/proc 的内容写到/tmp/proc 文件里，每个月一次（ls /proc >> /tmp/proc）。

6.8.5. 配置 Cron 任务

Cron 任务的配置文件是/etc/crontab 文件，它只能通过 root 用户进行访问。该文件包含以下内容：

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
```

```
MAILTO=root
HOME=/
# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

前三行包含了和 anacrontab 文件一样的变量定义：SHELL、PATH 和 MAILTO。要了解更多有关这几个变量的信息，请参见 6.8.4 配置 Anacron 任务。

此外，该文件还可以定义 HOME 变量。HOME 变量定义一个目录，该目录在任务执行命令或脚本时将被作为家目录。

/etc/crontab 文件的剩余行代表计划任务，并遵从以下格式：

```
minute hour day month day of week username command
```

以下各项定义了任务将要运行的时间：

- minute——从 0 到 59 的任意整数
- hour——从 0 到 23 的任意整数
- day——从 1 到 31 的任意整数（如果指定了月份，则这里的日期必须是相对于该月有效的日期）
- month——从 1 到 12 的任意整数（或者月份的简短名称，例如 jan 或者 feb）
- day of week——从 0 到 7 的任意整数，0 或 7 代表星期日（或者使用每周各天的简短名称，例如 sun 或者 mon）

以下各项定义了任务的其它属性：

- username——指定执行任务的用户。
- command——将要执行的命令。

这里的命令既可以是一个类似 ls /proc >> /tmp/proc 的命令，也可以是一个执行自定义脚本的命令

对于以上的各项属性值，星号（*）可以用来表示所有的有效值。例如，如果您将 month 的值定义为星号，则该任务将在其他条件的约束下每个月都执行。

整数之间的连字号（-）代表了一个整数范围。例如，1-4 表示整数 1、2、3、4。

用逗号（,）分隔的值列表指定了一个列表。例如，3,4,6,8 就确实指明了这四个整数。

斜杠（/）可以用来指定步长值。指定了步长值的项将按步长所定义的整数来增长。例如，minute 项的值定义为 0-59/2，则表示每隔一分钟。步长值也可以和星号一起使用。例如，如果 month 项的值被定义为*/3，则任务将每三个月执行一次（每隔两个月）。

以井号（#）开头的所有行都是注释，不会被处理。

非 root 用户可以使用 crontab 工具来配置 cron 任务。用户定义的 crontabs 被保存在/var/spool/cron/目录下，并且以创建它们的用户来执行。

要以一个特定用户来创建 crontab，请以该用户进行登录，并且输入命令 crontab -e 来使用 VISUAL 或者 EDITOR 环境变量所指定的编辑器对用户的 crontab 进行编辑。该文件使用和/etc/crontab 完全相同的格式。当用户对 crontab 的修改被保存时，crontab 按照用户名来保存，并写入/var/spool/cron/username 文件中。要列出当前用户的 crontab 文件的内容，使用 crontab -l 命令。

/etc/cron.d/目录下包含的文件和/etc/crontab 文件具有相同的语法。只有 root 用户被允许在该目录下创建和修改文件。

cron 守护进程会每分钟检查/etc/anacrontab 文件、/etc/crontab 文件、/etc/cron.d/目录和/var/spool/cron/目录的变化，并把检测到的修改加载到内存中。因此，当一个 anacrontab 或 crontab 文件被修改后，并不需要重启守护进程。

6.8.6. 控制对 Cron 的访问

要限制对 Cron 的访问，您可以使用/etc/cron.allow 和/etc/cron.deny 文件。这些访问控制文件使用相同的格式，都是每一行一个用户名。记住，两个文件都不允许使用空格。

如果存在 cron.allow 文件，只有在该文件中列出的用户才能使用 cron，并且 cron.deny 文件将被忽略。

如果 cron.allow 文件不存在，则 cron.deny 文件中列出的用户将被禁止使用 Cron。

如果访问控制文件被修改了，不必重启 Cron 守护进程（crond）。每次用户

试图添加或删除一个 cron 任务时，都会检查访问控制文件。

不管访问控制文件中的用户名是什么，root 用户都始终能够使用 cron。

您也可以通过插入式认证模块（Pluggable Authentication Modules, PAM）来控制访问。相应的设置保存在/etc/security/access.conf 文件中。例如，在文件中添加如下一行之后，将只有 root 用户能够创建 crontabs：

```
-:ALL EXCEPT root :cron
```

被禁止的任务将被记录在适当的日志文件中，或者，当使用 crontab -e 命令时，返回到标准输出里。要了解更多信息，请参考 access.conf.5 用户手册页面。

6.8.7. Cron 任务的黑白名单

任务的黑白名单用来定义任务的某部分不需要被执行。当在一个 Cron 目录里（例如，/etc/cron.daily/）调用 run-parts 脚本时这很有用：如果用户把这个目录下的程序加入了黑名单，run-parts 脚本将不会执行这些程序。

要创建一个黑名单，请在 run-parts 脚本将要执行的目录中创建一个 jobs.deny 文件。例如，如果您需要从/etc/cron.daily/目录中忽略一个特定的程序，请创建/etc/cron.daily/jobs.deny 文件。在这个文件中，指定需要在执行时被忽略的程序的名称（只有位于同一个目录下的程序能够加入列表）。如果一项任务执行一个命令，该命令从/etc/cron.daily/目录执行程序，例如 run-parts /ect/cron.daily，则 jobs.deny 文件中定义的程序不会被执行。

要定义一个白名单，请创建 jobs.allow 文件。

jobs.deny 和 jobs.allow 的使用规则和 6.8.6 控制对 Cron 的访问中描述的 cron.deny 和 cron.allow 的使用规则一样。

6.8.8. At 和 Batch

Cron 被用来调度重复性任务，At 工具被用来在一个指定的时间调度一次性任务，Batch 工具被用来调度将在系统平均负载低于 0.8 时被执行的一次性任务。

6.8.8.1. 安装 At 和 Batch

要确定您的系统上是否已经安装了 at 包，请执行以下命令：

```
rpm -q at
```

如果已经安装了，上述命令将返回 at 包的完整名称，否则将通知您包不可用。

要安装程序包，以 root 用户按照下面的格式来使用 yum 命令：

```
yum install package
```

例如，要同时安装 At 和 Batch，可以在命令行提示符下输入以下命令：

```
~]# yum install at
```

6.8.8.2. 运行 At 服务

At 和 Batch 任务都是由 atd 服务来控制的。本节将说明如何启动、停止和重启 atd 服务，以及如何配置让其开机自启动。

启动 At 服务

要确定服务是否正在运行，请使用以下命令：

```
systemctl status atd.service
```

要在当前会话中运行 atd 服务，请以 root 用户在命令行提示符下输入以下命令：

```
systemctl start atd.service
```

要配置服务开机自启动，请以 root 用户执行以下命令：

```
systemctl enable atd.service
```

建议您将 atd 服务在您的系统中配置为开机自启动。

停止 At 服务

要在当前会话中停止 atd 服务，请以 root 用户在命令行提示符下输入以下命令：

```
systemctl stop atd.service
```

要禁止服务开机自启动，请以 root 用户执行以下命令：

```
systemctl disable atd.service
```

重启 At 服务

要重启 atd 服务，请以 root 用户在命令行提示符下输入以下命令：

```
systemctl restart atd.service
```

该命令停止服务后将很快地再次启动服务。

6.8.8.3. 配置 At 任务

要使用 At 工具调度一次性任务在指定时间执行，请按以下步骤操作：

1. 在命令行中输入命令 at TIME，这里的 TIME 表示命令执行的时间。

TIME 参数可以使用以下任意一种格式定义：

- HH:MM：指定确切的小时和分钟，例如，04:00 表示上午 4 点。
- midnight：指定为午夜 12 点。
- noon：指定为中午 12 点。
- teatime：指定为下午 4 点。
- MONTHDAYYEAR 格式：例如，January 15 2020 表示 2020 年 1 月 15 日。年份的值是可选的。
- MMDDYY、MM/DD/YY 或者 MM.DD.YY 格式：例如，011520 表示 2020 年 1 月 15 日。
- now + TIME：这里的 TIME 由一个整数和 minutes、hours、days 或者 weeks 几个类型值来定义。例如，now + 5 days 表示命令将在从现在起五天后的同一时间被执行。

必须首先指定时间，其后可以指定可选的日期。要了解更多关于时间格式的信息，请参考/usr/share/doc/at-<version>/timespec 文本文件。

如果指定的时间已经过了，则任务将在明天的同一时间被执行。

2. 在显示出的 at>命令行提示符下，定义任务命令：

A. 输入任务应该执行的命令，并按下回车键。可选地，可以重复此步骤，提供多个命令。

B. 在命令行提示符下输入一个 shell 脚本，并在脚本的每一行之后按下回车键。

任务将使用用户的 SHELL 环境变量所设置的 shell、用户的登录 shell，或者 /bin/sh（无论先找到哪一个）。

3. 一旦输入结束，请在一个空行中按下 Ctrl+D 组合键，退出命令行提示符。

如果这一系列命令或者脚本试图在标准输出中显示信息，则输出将会以电子邮件的形式发送给用户。

要查看等待执行的任务列表，请使用 atq 命令。请参见 6.8.8.5 查看等待执行的任务了解更多信息。

您也可以限制 at 命令的使用。要了解更多信息，请参见 6.8.8.7 控制对 At 和 Batch 的访问。

6.8.8.4. 配置 Batch 任务

Batch 程序在系统平均负载降低到 0.8 以下的时候执行已定义的一次性任务。

要定义 Batch 任务，请按以下步骤进行操作：

1. 在命令行中输入 batch 命令。

2. 在显示出的 at>命令行提示符下，定义任务命令：

A. 输入任务应该执行的命令，并按下回车键。可选地，可以重复此步骤，

提供多个命令。

B. 在命令行提示符下输入一个 shell 脚本，并在脚本的每一行之后按下回车键。

任务将使用用户的 SHELL 环境变量所设置的 shell、用户的登录 shell，或者/bin/sh（无论先找到哪一个）。

3. 一旦输入结束，请在一个空行中按下 Ctrl+D 组合键，退出命令行提示符。

如果这一系列命令或者脚本试图在标准输出中显示信息，则输出将会以电子邮件的形式发送给用户。

要查看等待执行的任务列表，请使用 atq 命令。请参见 6.8.8.5 查看等待执行的任务了解更多信息。

您也可以限制 batch 命令的使用。要了解更多信息，请参见 6.8.8.7 控制对 At 和 Batch 的访问。

6.8.8.5. 查看等待执行的任务

要查看等待执行的 At 和 Batch 任务，可以执行 atq 命令。atq 命令将显示一个等待执行的任务的列表，每个任务单独显示为一行。每一行的格式为任务编号、日期、任务类型和用户名称。用户只能查看他们自己的任务。如果 root 用户执行 atq 命令，则会显示所有用户的所有任务。

6.8.8.6. 额外的命令行选项

at 和 batch 命令的额外命令行选项如下所示：

表格 6-7 at 和 batch 命令行选项

选项	描述
-f	从一个文件中读取命令或 shell 脚本，而不是在命令行提示符下输入命令或脚本。
-m	当任务完成后向用户发送电子邮件。
-v	显示任务被执行的时间。

6.8.8.7. 控制对 At 和 Batch 的访问

您可以使用/etc/at.allow 和/etc/at.deny 文件来限制对 at 和 batch 命令的访问。这些访问控制文件使用相同的格式，都是每一行一个用户名。记住，两个文件都不允许使用空格。

如果存在 at.allow 文件，只有在该文件中列出的用户才能使用 at 或者 batch，并且 at.deny 文件将被忽略。

如果 at.allow 文件不存在，则 at.deny 文件中列出的用户将被禁止使用 at 或者 batch。

如果访问控制文件被修改了，不必重启 at 守护进程（atd）。每次用户试图执行 at 或 batch 命令时，都会读取访问控制文件。

不管访问控制文件中的内容是什么，root 用户都始终能够执行 at 和 batch 命令。

6.9. 自动程序错误报告工具（ABRT）

6.9.1. ABRT 简介

自动程序错误报告工具，通常简写为 ABRT，是一个设计用于帮助用户检测和报告应用程序崩溃的工具集。它的主要目的是简化报告问题和查找解决方案的过程。

ABRT 由 abrtd 守护进程和许多用于处理、分析和报告检测到的问题的系统服务和工具组成。守护进程大多数时间都静默地在后台运行，当检测到一个应用程序崩溃或者内核故障时才活跃起来。然后守护进程搜集相关的问题数据，例如核心文件（如果有的话）、崩溃应用程序的命令行参数，以及其它数据。

ABRT 目前支持对 C、C++、Java、Python 和 Ruby 所编写的应用程序崩溃的检测，以及对 X.Org 崩溃、内核故障和内核严重错误的检测。要了解更多有关所支持的故障和崩溃类型的信息，以及检测众多类型的崩溃的方式，请参见 6.9.4 检测软件问题。

处理已存在的问题数据（与创建新的问题数据不同）的 ABRT 组件是一个名为 libreport 的独立项目的一部分。libreport 库提供了一个分析和报告问题的通用机制，它不仅被 ABRT 使用，也同样被应用程序所使用。然而，ABRT 和 libreport 的操作和配置已经紧密地整合在一起了，因此，在本文档中将作为一个整体进行讨论。

6.9.2. 安装 ABRT 并启动服务

为了使用 ABRT，请确保您的系统中已经安装了 abrt-desktop 包或者 abrt-cli 包。abrt-desktop 安装包为 ABRT 提供了一个图形化的用户界面，abrt-cli 则包含一个在命令行使用 ABRT 的工具。您可以同时安装两个包。ABRT 图形用户界面和命令行工具的通用工作流是相似的，并且遵从相同的模式。

⚠ 注意，安装 ABRT 软件包将会覆写/proc/sys/kernel/core_pattern 文件，该文件包含了用于命名内核转储文件的模板。该文件的内容将被覆写为：

```
/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e %P %I
```

参见 2.2.4 安装软件包了解如何在使用 Yum 包管理器安装软件包。

6.9.2.1. 安装 ABRT 图形用户界面

ABRT 图形用户界面提供了一个在桌面环境中易于使用的前端工具。您可以

作为 root 用户来执行以下命令安装所需要的包：

```
~]# yum install abrt-desktop
```

安装之后，ABRT 通知程序将被配置为当您的图形桌面会话启动后自动启动。您可以通过在终端中执行以下命令来验证 ABRT 小程序是否正在运行：

```
~]$ ps -el | grep abrt-applet
0 S 1000 23029 22660 0 80 0 - 112932 poll_s ? 00:
00:00 abrt-applet
```

如果小程序没有运行，您可以通过运行 abrt-applet 程序来在您当前的桌面会话中手动启动它：

```
~]$ abrt-applet &
[1] 2261
```

6.9.2.2. 安装 ABRT 命令行工具

命令行界面对于没有图形界面的机器、通过网络连接的远程系统或者在脚本中，是很有用的。您可以作为 root 用户来执行以下命令安装所需要的包：

```
~]# yum install abrt-cli
```

6.9.2.3. 安装 ABRT 附属工具

要接收 ABRT 检测到的关于崩溃的电子邮件通知，您需要安装 libreport-plugin-mailx 软件包。您可以作为 root 用户来执行以下命令安装所需要的包：

```
~]# yum install libreport-plugin-mailx
```

默认情况下，它将在本地机器上给 root 用户发送通知。电子邮件目的地可以在/etc/libreport/plugins/mailx.conf 文件中进行配置。

要在登录时将通知显示在您的控制台上，请安装上 abrt-console-notification 软件包。

ABRT 能够检测、分析并报告多种类型的软件故障。默认情况下， ABRT 安装支持最通用的故障类型，例如 C 和 C++应用程序的崩溃。对其它故障类型的支持通过独立软件包来提供。例如，要安装支持对 Java 语言编写的应用程序中的异常的检测，请以 root 用户执行以下命令：

```
~]# yum install abrt-java-connector
```

参见 6.9.4 检测软件问题获取 ABRT 所支持的预研和软件项目列表。该节也包括了一个要启用多种故障类型的检测所需要安装的对应的包的列表。

6.9.2.4. 启动 ABRT 服务

abrtd 守护进程被配置为开机自启动。您可以使用以下命令来验证它当前的状态：

```
~]$ systemctl is-active abrtd.service
active
```

如果 systemctl 命令返回了 inactive 或者 unknown，则表示守护进程没有运行。您可以作为 root 用户执行以下命令来在当前会话中启动它：

```
~]# systemctl start abrtd.service
```

类似地，您可以通过相同的步骤来检查处理各种故障类型的服务的运行状态，并启动它们。例如，如果您希望 ABRT 能够检测 C 或者 C++崩溃，请确保 abrt-ccpp 服务正在运行。请参见 6.9.4 检测软件问题，获取所有可用的 ABRT 检测服务及它们各自对应的安装包的列表。

除了 abrt-vmcore 和 abrt-pstoreoops 服务是在内核严重故障或内核故障实际发生时才被启动外，所有的其它 ABRT 服务都是在它们各自的安装包被安装时，就被设置为开机启动，并立即启动的。就如 3.1 使用 systemd 管理系统服务所描述的那样，您可以使用 systemctl 工具禁用或启用任何 ABRT 服务。

6.9.2.5. 测试 ABRT 崩溃检测

要测试 ABRT 是否能正确地工作，可以使用 kill 命令发送一个 SEGV 信号来终止一个进程。例如，按照下述方式启动一个 sleep 进程，并使用 kill 命令终止该进程：

```
~]$ sleep 100 &
[1] 2823
~]$ kill -s SEGV 2823
```

ABRT 在执行 kill 命令之后，很快就检测到了崩溃的发生，假若正在运行图形会话，用户将会被图形用户界面通知小程序通告检测到的问题。在命令行环境中，您可以使用 abrt-cli list 命令来检查是否检测到了崩溃的发生，或者通过检查在/var/tmp/abrt/目录里创建的崩溃转储来确定。请参见 6.9.5 处理检测到的问题了解更多有关如何处理检测到的崩溃的信息。

6.9.3. 配置 ABRT

在 ABRT 中，一个问题的生命周期是通过事件来驱动的。例如：

- 事件#1——创建问题数据目录。
- 事件#2——分析问题数据。

- 事件#3——报告问题。

无论何时检测到一个问题之后，ABRT 将该问题和所有已存在的问题数据进行比较，确定是否是已经被记录的相同的问题。如果是相同的问题，则更新已存在的问题数据，并且最近的（重复的）问题不会被再次记录。如果不是相同的问题，则会创建一个问题数据目录。一个问题数据目录通常由以下文件组成：analyzer、architecture、coredump、cmdline、executable、kernel、os_release、reason、time 和 uid。

其它文件，例如 backtrace，将会在问题的分析过程中被创建，这取决于所使用的分析方法和它的配置设定。这些文件中的每一个都各自保存着有关系统和问题本身的特定信息。例如，kernel 文件里记录了崩溃内核的版本。

在创建了问题数据目录，并收集了问题数据之后，您可以通过 ABRT 图形用户界面或者 abrt-cli 命令行工具来处理问题。请参见 6.9.5 处理检测到的问题了解更多关于 ABRT 所提供的处理已记录问题的工具的信息。

6.9.3.1. 配置事件

ABRT 事件使用插件来实现实际的报告操作。插件是事件用来调用处理问题数据目录的小工具。通过使用插件，ABRT 能够向不同的目的地报告问题，而几乎每一个报告目的地都需要一些配置。例如，Bugzilla 需要一个用户名、密码，以及一个指向 Bugzilla 服务实例的网址。

一些配置细节可以使用默认值（例如 Bugzilla 网址），但是其他配置细节则没有合理的默认值（例如用户名）。ABRT 在配置文件中查找这些设定，例如 report_Bugzilla.conf，该文件在/etc/libreport/events/或者$HOME/.cache/abrt/events/目录中分别代表系统范围的设置和特定用户的设置。配置文件中包含了成对的指令和值。

这些文件是运行事件和处理问题数据所必需的。gnome-abrt 和 abrt-cli 工具从这些文件中读取配置数据，并传递给它们所运行的事件。

关于事件的额外信息（例如它们的描述、名称、能够作为环境变量传递给它们的参数类型，以及其它属性）保存在/usr/share/libreport/events/目录下的 event_name.xml 文件里。gnome-abrt 和 abrt-cli 都使用了这些文件，以使用户界面更友好。请不要编辑这些文件，除非您想改变标准安装。如果您确实打算这样做，请将要修改的文件复制到/etc/libreport/events/目录下，并修改新复制的文件。这些文件可能包含以下信息：

- 易于使用的事件名称和描述
- 问题数据目录中需要事件处理的条目列表
- 要发送或者不发送的条目的默认和强制选择

- 图形用户界面是否应该提示数据审查
- 存在的配置选项，它们的类型（string、Boolean 等）、默认值、提示字符串等；这可以让图形用户界面构造出适当的配置对话框

例如，report_Logger 事件可以接受一个输出文件名作为参数。通过使用各自的 event_name.xml 文件，ABRT 图形用户界面可以确定哪些参数可以指定给一个选定的事件，并允许用户为这些参数设置参数值。这些值被 ABRT 图形用户界面保存，并在后续对这些事件的调用中重用。注意，ABRT 图形用户界面使用 GNOME Keyring 工具来保存配置选项，并且将它们传递给事件，这将覆盖从文本配置文件中获取的数据。

要打开图形化的【配置】窗口，请在 gnome-abrt 应用程序的运行实例中选择【自动程序错误报告工具】→【首选项】。这个窗口会显示在使用图形用户界面的报告过程中，可以被选择事件列表。当您选中了其中一个可配置的事件时，您可以点击【配置】按钮，为事件修改设定。

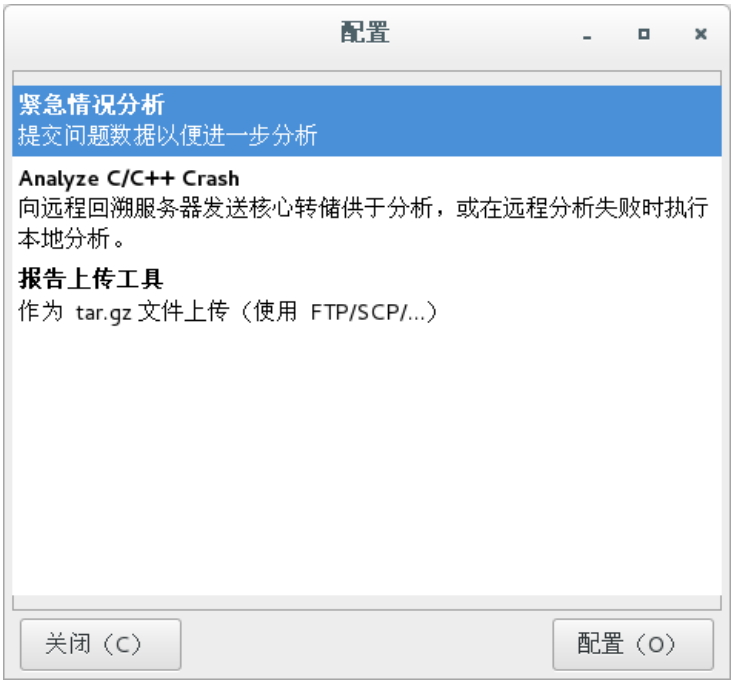


图 6-11 配置 ABRT 事件

/etc/libreport/目录层级下的所有文件都是全面可读的，被用来作为全局设置。因此，不建议在其中保存用户名、密码或者任何其它敏感数据。针对用户的设置（在 GUI 应用程序中设置，只能被$HOME 的所有者读取）安全地保存在 GNOME Keyring 中，如果使用的是 abrt-cli，这些设置也可以保存在$HOME/.abrt/目录下的文本配置文件中。

下面的表格显示了 ABRT 标准安装所提供的对默认的分析、收集和报告事

件的选择。表格中列出了每个事件的名称、标识符、/etc/libreport/events.d/目录下的配置文件，以及一个简要的描述。注意，由于配置文件使用的是事件标识符，ABRT 图形用户界面使用它们的名称来引用各个事件。另外需要注意的是，不是所有的事件都可以使用图形用户界面进行设置。要了解如何定义自定义事件，请参考“创建自定义事件”章节。

表格 6-8 标准 ABRT 事件

名称	标识符和配置文件	描述
uReport	report_uReport	上传一个 μReport 到 FAF 服务器上。
Mailx	report_Mailx mailx_event.conf	通过 Mailx 工具把问题报告发送到一个指定的电子邮件地址。
Bugzilla	report_Bugzilla bugzilla_event.conf	将问题报告给指定的 Bugzilla。
Emergency analysis	report_EmergencyAnalysis emergencyanalysis_event.conf	上传一个压缩包到 FAF 服务器上进行进一步分析。在标准报告方法失败时使用。
Analyze C or C++ Crash	analyze_CCpp ccpp_event.conf	将核心转储发送到一个远程追踪服务器上进行分析，或者远程失效的话执行本地分析。
Report uploader	report_Uploader uploader_event.conf	使用 FTP 或 SCP 协议将包含问题数据的压缩归档文件（.tar.gz）上传到所选的目的地。
Analyze VM core	analyze_VMcore vmcore_event.conf	在内核故障的问题数据上运行 GDB（GNU 调试器），生成一个内核回溯（backtrace）。
Local GNU Debugger	analyze_localGDB ccpp_event.conf	在应用程序的问题数据上运行 GDB（GNU 调试器），生成一个程序回溯（backtrace）。
Collect.xsession-errors	analyze_xsession_errors ccpp_event.conf	将~/.xsession-errors 文件的相关行保存到问题报告中。
Logger	report_Logger	创建一个问题报告，并将其保存到

名称	标识符和配置文件	描述
	print_event.conf	一个指定的本地文件中。
Kerneloops.org	report_Kerneloops koops_event.conf	将内核问题发送到 kerneloops.org 上的故障追踪系统。

6.9.3.2. 创建自定义事件

每一个事件都是在分别的配置文件中通过一个规则结构来定义的。配置文件通常保存在/etc/libreport/events.d/目录下。这些配置文件都是通过主配置文件/etc/libreport/report_event.conf 来加载的。该文件可以接受 shell 元字符（*、$、？ 等），并可以相对于其位置来解释相对路径。

每一个规则以一个非空格前导字符行开始，随后的所有以空格符或制表符开始的行都被认为是这个规则的一部分。每一个规则由两个部分组成，一个条件部分和一个程序部分。条件部分包含以下列形式之一来表示的条件：

- VAR=VAL
- VAR!=VAL
- VAL~=REGEX

在这里：

- VAR 要么是 EVENT 关键字，要么是一个问题数据目录元素的名称（例如 executable、package、hostname 等）
- VAL 要么是一个事件名称，要么是一个问题数据元素
- REGEX 是一个正则表达式

程序部分由程序名称和 shell 可解释代码组成。如果条件部分的所有条件都是有效的，则程序部分将在 shell 中运行。以下是一个事件的示例：

```
EVENT=post-create date > /tmp/dt
    echo $HOSTNAME `uname -r`
```

这个事件将会用当前的日期和时间重写/tmp/dt 文件的内容，并将机器的主机名和它的内核版本打印到标准输出中。

接下来是一个更复杂的事件的示例，它实际是预先定义的事件之一。它将~/.xsession-errors 文件的相关行保存到 abrt-ccpp 服务用来处理的任何问题的问题报告中，这里假定崩溃的应用程序在崩溃时加载过任意 X11 库：

```
EVENT=analyze_xsession_errors analyzer=CCpp dso_list~=.*/libX11.*
    test -f ~/.xsession-errors || { echo "No ~/.xsession-errors"; exit
```

```
1; }
        test -r ~/.xsession-errors || { echo "Can't read ~/.xsession-errors
"; exit 1; }
        executable=`cat executable` &&
        base_executable=${executable##*/} &&
        grep -F -e "$base_executable" ~/.xsession-errors | tail -999 >xs
ession_errors &&
        echo "Element 'xsession_errors' saved"
```

可能的事件集没有被限定。系统管理员可以根据他们的需要添加事件。目前，标准的 ABRT 和 libreport 安装提供了以下事件名称：

post-create

这个事件被 abrtd 用来处理新创建的问题数据目录。当 post-create 事件运行时，abrtd 检查新的问题数据的 id 是否和任何已存在的问题目录的 id 匹配。如果存在这样一个问题目录，则新的问题数据将被删除。

notify，notify-dup

notify 事件在 post-create 完成之后运行。当该事件运行时，用户可以确定问题是值得他们注意的。notify-dup 是类似的，只不过它用于相同问题的重复发生。

analyze_name_suffix

这里的 name_suffix 是事件名称的可替换部分。该事件用来处理收集的数据。例如，analyze_LocalGDB 事件使用 GNU 调试器（GDB）工具来处理一个应用程序的核心转储，并生成崩溃的回溯。

collect_name_suffix

这里的 name_suffix 是事件名称的可替换部分。该事件用来收集问题的额外信息。

report_name_suffix

这里的 name_suffix 是事件名称的可替换部分。该事件用来报告问题。

6.9.3.3. 设置自动报告

ABRT 可以被配置来自动发送任何检测到的问题或者崩溃的最初的匿名报告或者 uReports，而不用任何用户干预。当自动报告被开启后，通常在崩溃报告过程的开始阶段发送的所谓的 uReport，将在检测到崩溃后立即发送。这可以防止基于相同的崩溃进行重复支持。要启用自动报告功能，请以 root 用户执行以下命令：

```
~]# abrt-auto-reporting enabled
```

以上命令将/etc/abrt/abrt.conf 配置文件中的 AutoreportingEnabled 指令设置为 yes。这一系统范围的设定将应用到系统中的所有用户。注意，通过启用该选项，自动报告在图形桌面环境中也将被启用。要只启用 ABRT 图形用户界面中的自动报告，请在【配置错误报告】窗口中将【自动发送 uReport】选项的开关切换为【开启】。要打开这个窗口，请在 gnome-abrt 应用程序的运行实例中选择【自动程序错误报告工具】→【ABRT 配置】。要运行该应用程序，请选择【应用程序】→【杂项】→【自动程序错误报告工具】。执行 system-config-abrt 命令，打开配置错误报告界面，如图。

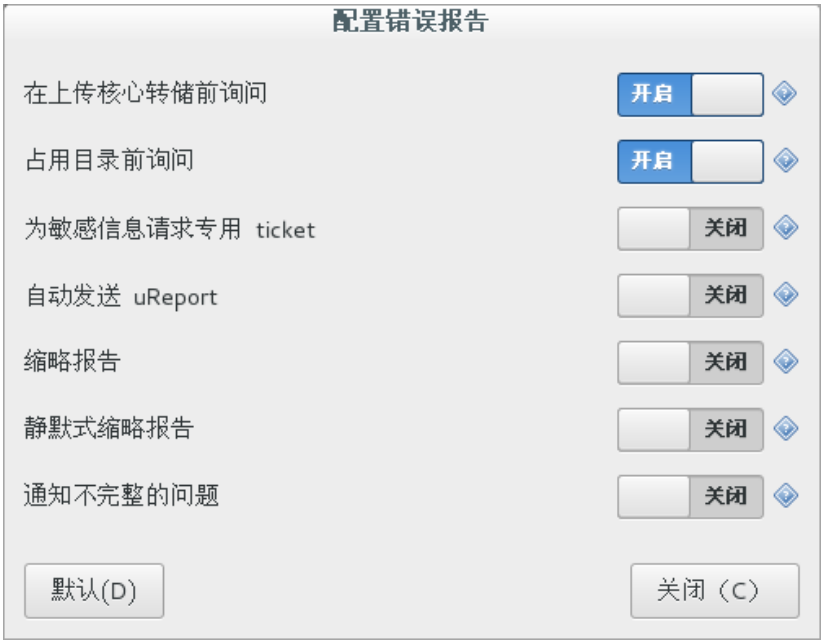


图 6-12 配置 ABRT 问题报告

uReport（微报告）是表示一个诸如二进制崩溃或者内核故障问题的 JSON 对象。这些报告设计得很简要、机器可读，并且完全匿名，这也是为什么它们能用于自动报告。uReports 使得对缺陷发生的追踪成为可能，但是它们通常不能提供足够的信息给工程师修复缺陷。

6.9.4. 检测软件问题

ABRT 能够检测、分析和处理使用多种不同的编程语言编写的应用程序崩溃。许多支持检测不同类型的崩溃的软件包，在安装主要的 ABRT 软件包（abrt-desktop、abrt-cli）时，就已经被安装上了。查看下面的表格了解所支持的崩溃类型和各自对应的软件包。

表格 6-9 支持的编程语言和软件项目

语言/项目	安装包
C 或者 C++	abrt-addon-ccpp
Python	abrt-addon-python
Ruby	rubygem-abrt
Java	abrt-java-connector
X.Org	abrt-addon-xorg
Linux（内核故障）	abrt-addon-kerneloops
Linux（内核严重故障）	abrt-addon-vmcore
Linux（永久存储）	abrt-addon-pstoreoops

6.9.4.1. 检测 C 和 C++崩溃

abrt-ccpp 服务安装了它自己的核心转储处理程序，当该服务被启动后，将会覆盖内核的 core_pattern 变量的默认值，从而让 abrtd 来处理 C 和 C++崩溃。如果您停止了 abrt-ccpp 服务，之前指定的 core_pattern 变量值将重新恢复为默认值。

默认情况下，/proc/sys/kernel/core_pattern 文件的内容包含了字符串 core，意味着内核将在崩溃进程的当前目录下生成带有“core.”前缀的文件。abrt-ccpp 服务将重写 core_pattern 文件的内容，使其包含如下命令：

```
/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e %P %I
```

该命令指示内核将核心转储用管道连接至 abrt-hook-ccpp 程序，该程序会将其保存到 ABRT 的转储位置，并通知 abrtd 进程有新的崩溃。出于调试的目的，它也会从/proc/PID/目录中保存以下文件：maps、limits、cgroup、status。请参见 proc(5)获取有关格式的描述和这些文件的含义。

6.9.4.2. 检测 Python 异常

abrt-addon-python 安装包为 Python 应用程序安装了一个自定义的异常处理程序。Python 解释器将会自动导入安装到/usr/lib64/python2.7/site-packages/目录下的 abrt.pth 文件，abrt.path 文件又会导入 abrt_exception_handler.py。这样，Python 的默认的 sys.excepthook 将被自定义的处理程序覆盖，处理程序通过它的 Socket API 将未处理的异常转发给 abrtd。

要禁用自动导入特定位置的模块，从而阻止 Python 程序在运行时使用 ABRT 自定义的异常处理程序，请传递-S 选项给 Python 解释器：

```
~]$ python -S file.py
```

在上述命令中，请将 file.py 替换为在不使用特定位置模块的情况下您想执行的 Python 脚本的名称。

6.9.4.3. 检测 Ruby 异常

rubygem-abrt 安装包使用 at_exit 特性注册了一个自定义的处理程序，该处理程序将在一个程序结束时被执行。这样可以允许对未处理异常进行检查。每次捕获到一个未处理异常时，ABRT 处理程序就会准备一个缺陷报告，该报告可通过标准的 ABRT 工具进行提交。

6.9.4.4. 检测 Java 异常

ABRT Java 连接器是一个将未捕获的 Java 异常报告给 abrtd 的 JVM 代理。此代理注册了多个 JVMTI 事件回调函数，并通过使用-agentlib 命令行参数加载到 JVM 中。注意，处理注册的回调函数会影响应用程序的性能。可以使用以下命令来让 ABRT 从一个 Java 类中捕获异常：

```
~]$ java -agentlib:abrt-java-connector[=abrt=on] $MyClass -platform.jvmtiSupported true
```

在上述命令中，使用您想要测试的 Java 类的名称来替换$MyClass。通过传递“abrt=on”选项给连接器，您可以确保异常将交由 abrtd 来处理。如果您想让连接器将异常输出到标准输出中，请省略该选项。

6.9.4.5. 检测 X.Org 崩溃

abrt-xorg 服务从/var/log/Xorg.0.log 文件中收集并处理关于 X.Org 服务崩溃的信息。注意，如果加载了被加入黑名单的 X.org 模块，则不会生成报告。取而代之，将会在问题数据目录中创建一个带有适当解释的不可报告文件。您可以在/etc/abrt/plugins/xorg.conf 文件中找到一个黑名单模块的列表。默认情况下，只有专有的图形驱动模块会被加入黑名单。

6.9.4.6. 检测内核故障和严重故障

通过检查内核日志的输出，ABRT 能够捕获和处理所谓的内核故障——Linux 内核的正确行为的非致命偏差。此功能由 abrt-oops 服务提供。

ABRT 也能够使用 abrt-vmcore 服务来检测和处理内核严重故障——致命的、不可恢复的错误，需要重启系统来解决。该服务只有当在/var/crash/目录中出现 vmcore 文件时才会启动。当找到核心转储文件时，abrt-vmcore 将在/var/tmp/abrt/目录中创建一个新的问题数据目录，并将核心转储文件移动到新创建的问题数据目录中。在搜索完/var/crash/目录之后，该服务被停止。

为了让 ABRT 能够检测内核严重故障，必须在系统中启用 kdump 服务。必须正确地设置好为 kdump 内核预留的内存量。您可以使用 system-config-kdump 图形化工具来设置它。

在支持 pstore 的系统上，通过使用 abrt-pstoreoops 服务，ABRT 能够收集和处理有关内核严重故障的信息，该信息保存在自动挂载的/sys/fs/pstore/目录下。和平台相关的 pstore 接口（永久存储）提供了一种在系统重启时存储数据的机制，从而能够保留内核严重故障的信息。该服务将在/sys/fs/pstore/目录中出现内核崩溃转储文件时自动启动。

6.9.5. 处理检测到的问题

abrtd 所保存的问题数据可以被查看、报告和删除，您可以通过命令行工具 abrt-cli 或者图形化工具 gnome-abrt 来完成这些操作。

注意，ABRT 通过将新的问题和所有本地保存的问题进行比较，来识别重复问题。对于重复的崩溃，ABRT 只需要您对其操作一次。然而，如果您删除了那个问题的崩溃转储，那么下一次同样的问题发生时，ABRT 将把它作为新的崩溃来处理：ABRT 将会向您通告该问题，提示您填充描述信息，并报告该问题。为了避免 ABRT 向您通告重复发生的问题，请不要删除它的问题数据。

6.9.5.1. 使用命令行工具

在命令行环境中，用户在登录时被通告新的崩溃信息，这里假设用户已经安装了 abrt-console-notification 软件包。控制台通告看起来如下所示：

```
ABRT has detected 1 problem(s). For more info run: abrt-cli list --since 1398783164
```

要查看检测到的问题，请输入 abrt-cli list 命令：

```
~]$ abrt-cli list
```

在 abrt-cli list 命令的输出结果中列出的每一个崩溃，都有一个唯一标识符和一个目录，可以在使用 abrt-cli 做进一步的操作时用到它们。

要查看某个特定问题的信息，可以使用 abrt-cli info 命令：

```
abrt-cli info [-d] directory_or_id
```

要在使用 list 和 info 子命令时增加显示的信息量，请使用-d（--detailed）选项，该选项将会显示列出的问题的所有已保存的信息，包括各自的 backtrace 文件（如果已经生成了该文件的话）。

要分析并报告一个特定问题，请使用 abrt-cli report 命令：

```
abrt-cli report directory_or_id
```

如果您确定您不想报告某个特定的问题，您可以删除该问题。要删除一个问题，以便 ABRT 不再保留关于该问题的信息，请执行命令：

```
abrt-cli rm directory_or_id
```

要显示某个 abrt-cli 命令的帮助信息，请使用--help 选项：

```
abrt-cli command --help
```

6.9.5.2. 使用图形用户界面

无论何时一个问题报告被创建时，ABRT 守护进程将广播一个 D-Bus 消息。如果在一个图形化桌面环境中运行了 ABRT 小程序，它将捕获这个消息，并在桌面上显示一个通告对话框。您可以通过点击这个对话框上的【报告】按钮来打开 ABRT 的图形用户界面。您也可以通过选择【应用程序】→【杂项】→【自动程序错误报告工具】菜单项来打开 ABRT 的图形用户界面。

可选地，您可以在命令行中使用以下命令来运行 ABRT 的图形用户界面：

```
~]$ gnome-abrt &
```

ABRT 的图形用户界面窗口显示了一个已检测到问题的列表。每个问题条目由故障应用程序的名称、应用程序崩溃的原因，以及问题上一次发生的日期组成。

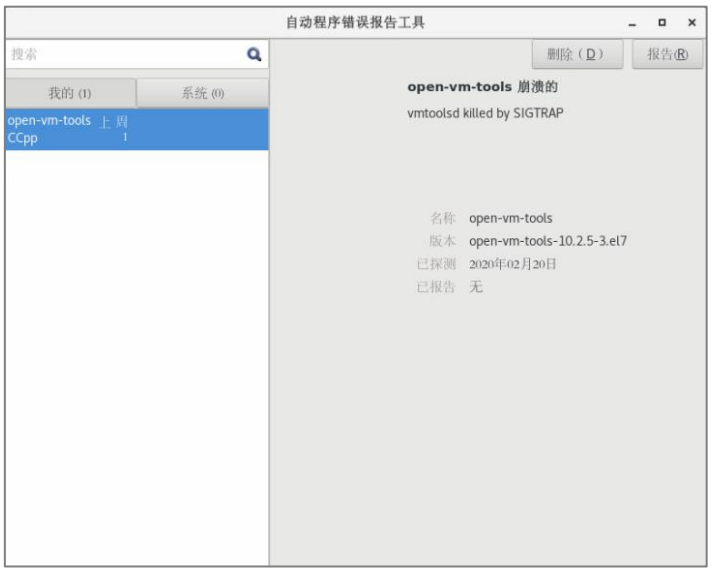


图 6-13 ABRT 图形用户界面

要访问更详细的问题描述，请双击要查看的问题或者选中列表中的问题后点击【报告】按钮。要删除一个问题，请点击【删除】按钮。

6.10. Oprofile

Oprofile 是一个低开销、系统范围的性能监视工具。它使用处理器上的性能监视硬件获取系统上关于内核和可执行程序的信息，例如当内存被引用的时候，L2 缓存的请求数，以及接收到的硬件中断数。在中标麒麟高级服务器操作系统软件中，要使用此工具必须安装上 oprofile 软件包。

许多处理器包含专用的性能监视硬件。该硬件使得检测什么时候发生了某些确定的事件成为可能（例如请求的数据不在缓存中）。该硬件通常表现为计数器的形式，每次一个事件发生时计数器的值就会增加。当计数器的值增加时，会产生一个中断，从而能够控制性能监视所产生的开销。

OProfile 使用这个硬件（如果没有性能监视硬件则使用一个基于计时器的替代品）来在每次计数器产生中断时收集性能相关的数据样本。这些取样将周期性地写入到磁盘中，之后，这些取样中所包含的数据能够被用来生成系统级和应用程序级的性能报告。

在使用 Oprofile 时请注意以下限制：

- 共享库的使用——共享库代码的取样没有被归属于特定的应用程序，除非使用了--separate=library 选项。
- 性能监视取样是不精确的——当一个性能监视寄存器触发一次取样时，不会像处理中断处理那样精确，例如：处理被零除这样的异常中断。由于处理器指令的无序执行，取样可能被记录在一个邻近的指令中。
- opreport 没有正确地为内联函数关联取样——opreport 使用一个简单的地址范围机制来确定地址中的函数。内联函数取样没有被归属于内联函数本身，而是被归属于了内联函数被插入的那个函数。
- OProfile 多次运行后累积了数据——OProfile 是一个系统范围的评测器，通常会预期进程将启动和停止多次。于是，取样将会因为多次运行而累积。可以使用命令 opcontrol --reset 来清除之前的取样。
- 硬件性能计数器不适用于虚拟机操作系统——因为硬件性能计数器在虚拟机操作系统中不可用，所以您需要使用 timer 模式。输入命令 opcontrol --dein it，然后执行 modprobe oprofile timer=1 来启用 timer 模式。
- 非 CPU 受限性能问题——OProfile 面向于查找 CPU 受限进程的问题。O Profile 不会去识别处于睡眠状态的进程，因为它们正在等待锁或者其它一些事件发生（例如等待一个 IO 设备完成操作）。

6.10.1. 工具概览

下表提供了 oprofile 安装包中最常用的工具的一个简要概览。

表格 6-10 常用命令

命令	描述
ophelp	显示系统处理器可用的事件，每个事件都有一个简要的描述。
opimport	将样本数据库文件从外部二进制格式转换为系统本地格式。只有在评测不同架构的样本数据库时需要使用此命令。
opannotate	如果应用程式是使用调试标志编译的，则为可执行文件创建带注释的源代码。
opcontrol	配置需要收集的数据。
operf	推荐用来替换 opcontrol 的评测工具。
opreport	获取评测数据。
oprofiled	以守护进程来运行，周期性地把取样数据写到磁盘。

6.10.1.1. operf 和 opcontrol

在使用 OProfile 收集评测数据时，有两种互斥的方法。您可以使用较新的、更推荐使用的 operf，也可以使用 opcontrol。

operf

这是推荐的评测模式。operf 工具使用 Linux 性能事件子系统，因此不需要 oprofile 内核驱动。operf 工具可以让您的性能评测更加精确，无论是对单个进程还是系统范围；也能够让 OProfile 和您系统上其它使用性能监视硬件的工具更好地共存。不像 opcontrol，operf 无需 root 权限就可以使用。不过，operf 在使用--system-wide 选项进行系统范围的操作时，仍然需要 root 授权。

operf 不需要最初的配置。您可以在调用 operf 时使用命令行选项来指定您的评测设定。之后，您可以运行在 6.10.6 分析数据中描述的后处理工具。请参见 6.10.2 使用 operf 了解更多信息。

opcontrol

此模式由 opcontrol shell 脚本、oprofiled 守护进程和几个后处理工具组成。opcontrol 命令用来配置、启动和停止一个评测会话。一个 OProfile 内核驱动（通常被构建为一个内核模块）被用来收集样本，这些样本被 oprofiled 记录在样本文件中。只有当您具有 root 权限时，您才可以使用此遗留模式。在某些确定的情况下，例如当您需要对禁用中断请求的区域取样时，此模式是一个比较好的替换方案。

6.10.2. 使用 operf

operf 是推荐使用的评测模式，它在启动前不需要初始化设置。所有的设置都以命令行选项的形式指定，它也没有单独的启动评测进程的命令。要停止 operf，请按 Ctrl+C 组合键。典型的 operf 命令语法如下所示：

```
operf options range command args
```

使用需要的命令行选项替换 options，来指定您的评测设置。在 operf(1)用户手册页面描述了完整的选项。使用以下之中的一个来替换 range：

--system-wide：该选项允许进行全局评测。

--pid=PID：该选项用来评测一个正在运行的应用程序，这里的 PID 是您想评测的进程的进程 ID。

通过 command 和 args，您可以定义一个要评测的命令或应用程序，以及该命令或应用程序需要输入的参数。command 需要--pid 或者--system-wide 选项，但是这两个选项不能同时使用。

当您在命令行中调用 operf 时不使用 range 选项，则将收集子进程的数据。

要运行 operf --system-wide，您需要 root 授权。当评测完成后，您可以使用 Ctrl+C 组合键来停止 operf。

如果您运行 operf --system-wide 时，是作为一个后台进程（使用了&）来运行的，需要以一种可控的方式来停止该进程，以便处理收集到的评测数据。要停止该后台进程，请使用以下命令：

```
kill -SIGINT operf-PID
```

在运行 operf --system-wide 命令时，建议您的当前工作目录应该是/root 或者/root 下的一个子目录，以便样本数据文件不会被存储在一个普通用户能够访问的位置。

6.10.2.1. 指定内核

要监视内核，请执行以下命令：

```
operf --vmlinux=vmlinux_path
```

使用这个选项，您可以指定一个和正在运行的内核相匹配的 vmlinux 文件的路径。内核取样将会被归属于这个二进制文件，从而允许后处理工具将样本归属于合适的内核符号。如果没有指定该选项，则所有的内核取样都将被归属于一个名为“no-vmlinux”的伪二进制文件。

6.10.2.2. 设定监视事件

大多数处理器包含了计数器，OProfile 用它们来监视特定的事件。如表格 6-1

2 OProfile 处理器和计数器所展示的，可用的计数器的数量取决于处理器。

每一个计数器的事件可以通过命令行或者图形界面来配置。要了解关于图形界面的更多信息，请参见 6.10.9 图形界面。如果计数器不能设定一个特定的事件，将会显示一个错误信息。

- 底层的 Linux 性能事件子系统内核不支持一些比较旧的处理器模型，因此 operf 也不支持这些处理器模型。如果您在尝试使用 operf 时接收到如下所示的消息：

```
Your kernel's Performance Events Subsystem does not support your processor type.
Please use the opcontrol command instead of operf.
```

可以试着用 opcontrol 来评测，看看也许 OProfile 的遗留模式能够支持您的处理器类型。

- 因为硬件性能计数器在虚拟机操作系统中不可用，您必须在虚拟机操作系统中启用 timer 模式来使用 operf。要启用 timer 模式，请以 root 用户执行以下命令：

```
opcontrol --deinit
```

```
modprobe oprofile timer=1
```

要通过命令行为每个可配置的计数器设定事件，请使用如下命令：

```
operf --events=event1,event2…
```

在这里，将为评测传递一个以逗号分隔的事件规范列表。每一个事件规范是一个以冒号分隔的属性列表，格式如下：

```
event-name:sample-rate:unit-mask:kernel:user
```

表格 6-11 事件规范对这些属性做了总结。最后三个属性是可选的，如果您省略了它们，它们将被设定为各自的默认值。注意，某些事件是需要单元掩码的。

表格 6-11 事件规范

属性	描述
event-name	从 ophelp 中获取到的确切的符号名称。
sample-rate	再次取样前需要等待的事件数。次数越小，取样就越频繁。对于不会频繁发生的事件，可能需要设置一个较小的值，以

属性	描述
	便在统计上能够捕获足够多的事件实例。另一方面，取样太频繁会使系统超载。默认情况下，OProfile 采用基于时间的事件设置，它会每个处理器每 100000 个时钟周期创建一个样本。
unit-mask	单元掩码用来进一步定义事件，可以用 ophelp 命令来列举事件的单元掩码。您可以使用一个以“0x”开头的十六进制值，或者和 ophelp 命令中的单元掩码描述中的名称相匹配的字符串来指定单元掩码。通过名称来定义只对那些在 ophelp 的输出结果中显示的那样，具有“extra:”参数的单元掩码有效。这一类的单元掩码不能通过十六进制值来定义。注意，在某些架构中，可能会有多个单元掩码具有相同的十六进制值。在那种情况下，它们只能通过名称来指定。
kernel	指定是否评测内核代码（值为 0 或 1，默认值为 1）。
user	指定是否评测用户空间代码（值为 0 或 1，默认值为 1）。

可用的事件根据处理器类型的不同而不同。如果没有指定事件规范，则将使用正在运行的处理器类型的默认事件来进行评测。请参见表格 6-13 默认事件来了解默认事件。要确定可用于评测的事件，请使用 ophelp 命令：

```
ophelp
```

6.10.2.3.样本分类

--separate-thread 选项通过线程组 ID（tgid）和线程 ID（tid）来对样本分类。这在多线程应用程序中查看每个线程的取样时很有用。当其和--system-wide 选项联合使用时，--separate-thread 也很有用，比如在一次评测运行的过程中，有多个进程在执行同一个程序，该选项就可以查看每个进程（每个线程组）的取样。

--separate-cpu 选项通过 CPU 来对样本分类。

6.10.3. 使用遗留模式配置 OProfile

OProfile 必须先加以配置才能以遗留模式运行。至少需要配置选择监视内核（或选择不监视内核）。下一小节描述如何使用 opcontrol 工具来配置 OProfile。当 opcontrol 命令被执行的时候，设定选项将被保存到/root/.oprofile/daemonrc 文件中。

6.10.3.1.指定内核

首先，配置 OProfile 是否应该监视内核。这是在启动 OProfile 前唯一需要配

置的选项。其它选项的配置都是可选的。

要监视内核，请以 root 用户执行以下命令：

```
opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -r`/vmlinux
```

为了监视内核，必须安装上包含了未压缩的内核的 kernel-debuginfo 安装包。

要配置 OProfile 不监视内核，请以 root 用户执行以下命令：

```
opcontrol --setup --no-vmlinux
```

该命令将会加载 oprofile 内核模块（如果该模块尚未加载的话），并创建/dev/oprofile/目录（如果该目录还不存在的话）。请参见 6.10.7 理解/dev/oprofile/目录了解有关该目录的细节。

设置样本是否应该在内核中收集只会改变所收集的数据，而不会改变收集数据的方法或贮存地点。

6.10.3.2. 设置要监视的事件

大多数处理器包含了计数器，OProfile 用它们来监视特定的事件。如表格 6-12 OProfile 处理器和计数器所展示的，可用的计数器的数量取决于处理器。

表格 6-12 OProfile 处理器和计数器

处理器	cpu_type	计数器数量
AMD64	x86-64/hammer	4
AMD Family 10h	x86-64/family10	4
AMD Family 11h	x86-64/family11	4
AMD Family 12h	x86-64/family12	4
AMD Family 14h	x86-64/family14	4
AMD Family 15h	x86-64/family15	6
Applied Micro X-Gene	arm/armv8-xgene	4
ARM Cortex A53	arm/armv8-ca53	6
ARM Cortex A57	arm/armv8-ca57	6
IBM eServer System i and IBM eServer System p	timer	1
IBM POWER4	ppc64/power4	8
IBM POWER5	ppc64/power5	6

处理器	cpu_type	计数器数量
IBM PowerPC 970	ppc64/970	8
IBM PowerPC 970MP	ppc64/970MP	8
IBM POWER5+	ppc64/power5+	6
IBM POWER5++	ppc64/power5++	6
IBM POWER6	ppc64/power6	6
IBM POWER7	ppc64/power7	6
IBM POWER8	ppc64/power8	8
IBM S/390 and IBM System z	timer	1
Intel Core i7	i386/core_i7	4
Intel Nehalem microarchitecture	i386/nehalem	4
Intel Westmere microarchitecture	i386/westmere	4
Intel Haswell microarchitecture (nonhyper-threaded)	i386/haswell	8
Intel Haswell microarchitecture (hyperthreaded)	i386/haswell-ht	4
Intel Ivy Bridge microarchitecture (nonhyper-threaded)	i386/ivybridge	8
Intel Ivy Bridge microarchitecture (hyperthreaded)	i386/ivybridge-ht	4
Intel Sandy Bridge microarchitecture (non-hyper-threaded)	i386/sandybridge	8
Intel Sandy Bridge microarchitecture	i386/sandybridge-ht	4
Intel Broadwell microarchitecture (nonhyper-threaded)	i386/broadwell	8
Intel Broadwell microarchitecture (hyperthreaded)	i386/broadwell-ht	4
Intel Silvermont microarchitecture	i386/silvermont	2
TIMER_INT	timer	1

使用表格 6-12 OProfile 处理器和计数器来确定您的 CPU 类型能够被同时监视的事件数量。如果处理器没有支持的性能监视硬件，计时器（timer）就会被用作处理器类型。

如果使用了计时器，就不能为任何处理器设置事件，因为硬件不支持硬件性能计数器。相反，计时器中断会被用来进行评测。

如果计时器没有被用作处理器类型，监视的事件就可以被改变，处理器的计数器 0 就会被默认设置为基于时间的事件。如果处理器上有多个计数器，0 以外的计数器默认不会被设置任何事件。被监视的默认事件显示在表格 6-13 默认事件中。

表格 6-13 默认事件

处理器	计数器的默认事件	描述
AMD Athlon and AMD 64	CPU_CLK_UNHALTED	处理器的时钟没有停止
AMD Family 10h, AMD Family 11h, AMD Family 12h	CPU_CLK_UNHALTED	处理器的时钟没有停止
AMD Family 14h, AMD Family 15h	CPU_CLK_UNHALTED	处理器的时钟没有停止
Applied Micro X-Gene	CPU_CYCLES	处理器周期
ARM Cortex A53	CPU_CYCLES	处理器周期
ARM Cortex A57	CPU_CYCLES	处理器周期
IBM POWER4	CYCLES	处理器周期
IBM POWER5	CYCLES	处理器周期
IBM POWER8	CYCLES	处理器周期
IBM PowerPC 970	CYCLES	处理器周期
Intel Core i7	CPU_CLK_UNHALTED	处理器的时钟没有停止
Intel Nehalem micro architecture	CPU_CLK_UNHALTED	处理器的时钟没有停止
Intel Pentium 4 (hyperthreaded and non hyper-threaded)	GLOBAL_POWER_EVENTS	处理器没有停止的时间
Intel Westmere micr	CPU_CLK_UNHALTED	处理器的时钟没有停止

处理器	计数器的默认事件	描述
oarchitecture		
Intel Broadwell microarchitecture	CPU_CLK_UNHALTED	处理器的时钟没有停止
Intel Silvermont microarchitecture	CPU_CLK_UNHALTED	处理器的时钟没有停止
TIMER_INT	（none）	每个计时器中断抽样

可以被同时监视的事件数量是由处理器的计数器数量决定的。不过，这不是一对一的关系；在一些处理器上，某些事件必须被映射到指定的计数器上。要确定可用的计数器数量，请执行以下命令：

```
ls -d /dev/oprofile/[0-9]*
```

可用的事件根据处理器类型的不同而不同。要确定可用于评测的事件，请使用 ophelp 命令。该命令返回的列表是针对系统处理器类型所特有的。

```
ophelp
```

如果 OProfile 没有被正确地配置，ophelp 命令将会失败，并返回以下错误信息：

```
Unable to open cpu_type file for reading
Make sure you have done opcontrol --init
cpu_type 'unset' is not valid
you should upgrade oprofile or force the use of timer mode
```

要配置 OProfile，请参见 6.10.3 使用遗留模式配置 OProfile。

每一个计数器的事件可以通过命令行或者图形界面来配置。要了解关于图形界面的更多信息，请参见 6.10.9 图形界面。如果计数器不能设定一个特定的事件，将会显示一个错误信息。

要通过命令行为每个可配置的计数器设置事件，请使用 opcontrol 命令：

```
opcontrol --event=event-name:sample-rate
```

把 event-name 替换成 ophelp 中显示的确切事件名称，把 sample-rate 替换为取样之间的事件数。

取样率

默认设置会选择基于时间的事件设置。它会每个处理器每 100000 个时钟周期创建一个样本。如果使用了计时器中断，计时器就被设置成两幅画面的最小时

间间隔率，而且还不能被用户设置。如果 cpu_type 不是 timer，就可以为每个事件设置了一个取样率。取样率是每次取样之间发生的事件数量。

在为计数器设置事件时，还可以指定一个取样率：

```
opcontrol --event=event-name:sample-rate
```

将 sample-rate 替换为再次取样前需要等待的事件数。次数越小，取样就越频繁。对于不会频繁发生的事件，可能需要设置一个较小的值，以便能够捕获足够多的事件实例。

⚠ 在设置取样率时请格外小心。取样太频繁会使系统超载，导致系统假死。

单元掩码

一些用户性能监视事件可能需要单元掩码来进一步定义事件。

每个事件的单元掩码能够通过 ophelp 命令列出。每个单元掩码的值以十六进制格式列出。要指定一个以上的单元掩码，十六进制值必须用按位或操作来组合。

```
opcontrol --event=event-name:sample-rate:unit-mask
```

注意，在某些架构中，可能会有多个单元掩码具有相同的十六进制值。在那种情况下，它们只能通过名称来指定。

6.10.3.3. 分离内核和用户空间评测

默认情况下，会为每个事件收集内核模式和用户模式的信息。要配置 OProfile 在某个指定的计数器中忽略内核模式的事件，请执行以下命令：

```
opcontrol --event=event-name:sample-rate:unit-mask:0
```

执行以下命令让计数器再次开始评测内核模式：

```
opcontrol --event=event-name:sample-rate:unit-mask:1
```

要配置 OProfile 在某个指定的计数器中忽略用户模式的事件，请执行以下命令：

```
opcontrol --event=event-name:sample-rate:unit-mask:1:0
```

执行以下命令让计数器再次开始评测用户模式：

```
opcontrol --event=event-name:sample-rate:unit-mask:1:1
```

当 OProfile 守护进程将评测数据写入样本文件时，它可以把内核和库评测数据分离到单独的样本文件中。要配置守护进程写入样本文件的方式，请以 root 用户执行以下命令：

```
opcontrol --separate=choice
```

choice 参数可以是以下之一：

- none——不要分离评测数据（默认值）。
- library——为库生成每个应用程序的评测数据。
- kernel——为内核和内核模块生成每个应用程序的评测数据。
- all——为库生成每个应用程序的评测数据，为内核和内核模块生成每个应用程序的评测数据。

如果使用了--separate=library，样本文件名在包括可执行文件名称的同时还包括库的名称。

配置信息的修改在 OProfile 评测器重新启动时生效。

6.10.4. 启动和停止 OProfile 遗留模式

要使用 OProfile 来开始监视系统，请以 root 用户执行以下命令：

```
opcontrol --start
```

所显示的输出和下面类似：

```
Using log file /var/lib/oprofile/oprofiled.log Daemon started. Profile
r running.
```

/root/.oprofile/daemonrc 中的设置将被使用。

OProfile 守护进程 oprofiled 被启动；它会周期性地把取样数据写到/var/lib/oprofile/samples/目录下。守护进程的日志文件位于/var/lib/oprofile/oprofiled.log。

在中标麒麟高级服务器操作系统软件 V7.0 中，nmi_watchdog 向 perf 子系统注册了。基于此，perf 子系统在启动时掌握了对性能计数器寄存器的控制权，使得 OProfile 不能正常工作。

要解决这个问题，要么以设置 nmi_watchdog=0 内核参数的方式来启动系统，要么在运行时以 root 用户来执行以下命令禁用 nmi_watchdog：

```
echo 0 > /proc/sys/kernel/nmi_watchdog
```

要重新启用 nmi_watchdog，请以 root 用户执行以下命令：

```
echo 1 > /proc/sys/kernel/nmi_watchdog
```

要停止评测器，请以 root 用户执行以下命令：

```
opcontrol --shutdown
```

6.10.5. 在遗留模式下保存数据

有时在一个指定的时间保存样本是很有用的。例如，在评测一个可执行文件时，根据不同的输入数据来收集不同的样本可能会很有用。如果要监视的事件数量超过了处理器可用的计数器数量，您可以运行多次 OProfile 来收集数据，每次都把样本数据保存到不同的文件中。

要保存当前的样本文件集合，请执行以下命令，把 name 替换成对当前会话的唯一描述性名称：

```
opcontrol --save=name
```

该命令会创建/var/lib/oprofile/samples/name/目录，并把当前的样本文件复制到该目录下。

要指定会话目录来保存样本数据，请使用--session-dir 选项。如果没有指定，则数据被保存到当前路径的 oprofile_data/目录下。

6.10.6. 分析数据

无论您使用 operf 还是遗留模式下的 opcontrol 来收集您的评测，它们都使用相同的 OProfile 后处理工具。

默认情况下，operf 将评测数据保存在当前目录的 oprofile_data/目录下。您可以使用--session-dir 选项来指定一个不同的位置。通常的评测后分析工具，例如 opreport 和 opannotate，可以用来生成评测报告。这些工具首先在当前目录的 oprofile_data/目录下搜索样本。如果不存在这个目录，分析工具将使用标准会话目录/var/lib/oprofile/。统计数据，例如接收到的总样本以及丢失样本，将被写入 session_dir/samples/operf.log 文件。

当使用遗留模式时，OProfile 守护进程 oprofiled，将会周期性地收集样本，并把它们写到/var/lib/oprofile/samples/目录下。在读取数据之前，请以 root 用户执行以下命令，以确保所有数据都被写入了该目录：

```
opcontrol --dump
```

每个样本文件都是基于可执行文件的名称来命名的。例如，在 Pentium III 处理器上对/bin/bash 进行默认事件评测的样本名称为：

```
\{root\}/bin/bash/\{dep\}/\{root\}/bin/bash/CPU_CLK_UNHALTED.100000
```

以下工具可以用来在收集好样本数据之后，对样本数据进行评测：

- opreport

- opannotate

用这些工具，以及被评测的二进制文件，可以生成能够被进一步分析的报告。

⚠ 被评测的可执行文件必须和这些工具一起使用，来分析数据。如果在收集数据之后必须修改可执行文件，请备份用来创建样本的可执行文件以及样本文件。注意，样本文件和二进制文件的名称必须一致。如果这些名称不匹配，您将无法做备份。作为替代方案，oparchive 可以用来处理这个问题。

每个可执行文件的样本被写入一个单独的样本文件中。从每个动态链接库中取得的样本也被写入一个单独的样本文件中。当 OProfile 运行时，如果被监视的可执行文件改变了，并且存在一个该可执行文件的样本文件，则已存在的样本文件将被自动删除。因此，如果已存在的样本文件是需要的，则必须对其进行备份，并且用来创建它的可执行文件，在被新版本的可执行文件替换之前也必须进行备份。OProfile 分析工具在分析时会使用生成样本的可执行文件。如果可执行文件改变了，分析工具将无法分析和其相关的样本。请参见 6.10.5 在遗留模式下保存数据了解如何备份样本文件。

6.10.6.1. 使用 opreport

opreport 工具可以提供被评测的所有可执行文件的一个概览。以下是 opreport 命令的部分样本输出：

```
~]$ opreport
```

每一个可执行文件都在列表中独占一行。第一列是为可执行文件所记录的样本数。第二列是可执行文件的样本数相对于总样本数所占的百分比。第三列是可执行文件的名称。

请参见 opreport(1)用户手册页面查看可用的命令行选项列表，例如-r 选项用来对输出结果按照可执行文件的样本数从小到大排序。您也可以使用-t 或者--threshold 选项来修剪 opcontrol 的输出结果。

6.10.6.2. 对单个可执行文件使用 opreport

要取回关于一个特定的可执行文件的更详细的评测信息，请使用：

```
opreport mode executable
```

请用将被分析的可执行文件的全路径来替换 executable。mode 表示以下选项中的一个：

-l

该选项用来按符号列出样本数据。例如，执行下面这个命令：

```
~]# opreport -l /lib/tls/libc-version.so
```

要按样本数从小到大（反序）对输出结果排序，可以让-r 选项和-l 选项结合使用。

-i symbol-name

该选项用来列出指定的符号名称的样本数据。例如，执行下面这个命令：

-d

该选项用来按符号列出样本数据，它比-l 选项的输出更详细。例如，执行下面这个命令：

```
~]# opreport -l -i __gconv_transform_utf8_internal /lib/tls/libc-version.so
```

-e symbol-name…

使用该选项，您可以在输出结果中排除一些符号。请用以逗号分隔的，您想排除的符号列表来替换 symbol-name。

session:name

这里，您可以指定会话的全路径，一个相对于/var/lib/oprofile/samples/的目录，或者如果您使用的是 operf，则是一个相对于./oprofile_data/samples/的目录。

6.10.6.3. 获取更详细的模块输出

OProfile 在系统范围内为运行在机器上的内核代码和用户空间代码收集数据。然而，一旦一个模块被加载到了内核中，关于该内核模块的起源信息就丢失了。该模块可能在启动时来自于 initrd 文件，可能来自于包含很多内核模块的目录，也可能来自于一个本地创建的内核模块。结果，当 OProfile 为一个模块记录样本时，它只是把该模块的样本列到了根目录下的一个可执行文件上，但这不可能是该模块实际代码的位置。您需要执行一些步骤来确保分析工具获取正确的可执行文件。

要获取模块动作的更详细视图，您可以安装上内核的 debuginfo 包。

请使用 uname -a 命令找出正在运行的内核，然后获取合适的 debuginfo 包，并将其安装到系统中。

之后请使用以下命令清理样本数据：

```
opcontrol --reset
```

要启动监视进程，例如，在一个 Westmere 处理器的机器上启动监视进程，可以使用以下命令：

```
~]# opcontrol --setup --vmlinux=/usr/lib/debug/lib/modules/`uname -r`/vmlinux --event=CPU_CLK_UNHALTED:500000
```

然后可以获取模块的详细信息，例如，可以使用以下命令获取 ext4 模块的详细信息：

```
~]# opreport /ext4 -l --image-path /lib/modules/`uname -r`/kernel
```

6.10.6.4. 使用 opannotate

opannotate 工具试图将特定指令的样本匹配到源代码的对应行上。最终生成的文件在左侧会显示对应行的样本。它也会在每个函数的前面添加一个注释，其中列出该函数的总样本。

要使用该工具，需要在系统中安装可执行文件的适当的 debuginfo 包。在中标麒麟高级服务器操作系统软件中，debuginfo 包不会跟随包含可执行文件的安装包而被自动安装。您需要单独获取并安装 debuginfo 包。

opannotate 命令的通用语法如下所示：

```
opannotate --search-dirs src-dir --source executable
```

这些命令行选项是强制的。请用一个包含源代码的目录路径替换 src-dir，并指定将要被分析的可执行文件。

6.10.7. 理解/dev/oprofile/目录

在使用 OProfile 遗留模式时，/dev/oprofile/目录用来为 OProfile 保存文件系统。另一方面，operf 不需要/dev/oprofile/。请使用 cat 命令来查看在这个文件系统中的虚拟文件的值。例如，以下命令显示 OProfile 检测到的处理器类型：

```
cat /dev/oprofile/cpu_type
```

在/dev/oprofile/目录中，每个计数器都存在一个对应的目录。例如，如果有 2 个计数器，则存在/dev/oprofile/0/和/dev/oprofile/1/两个目录。

每个计数器所对应的目录包含以下文件：

- count——取样间隔。
- enabled——如果值为 0，则计数器是关闭的，不会为其收集样本；如果值为 1，则计数器是打开的，将会为其收集样本。
- event——要监视的事件。
- extra——在具有 Nehalem 处理器的机器上用来进一步指定要监视的事件。
- kernel——如果值为 0，当处理器在内核空间时，就不会为这个计数器事件而收集样本；如果值为 1，即使处理器在内核空间时，也会收集样本。
- unit_mask——为计数器定义启用的单元掩码。
- user——如果值为 0，当处理器在用户空间时，就不会为这个计数器事件而收集样本；如果值为 1，即使处理器在用户空间时，也会收集样本。

这些文件的值可以用 cat 命令来获取，例如：

```
cat /dev/oprofile/0/count
```

6.10.7.1. 示例用法

OProfile 可以被开发者用来分析应用程序性能，也可以被系统管理员用来执行系统分析。例如：

- 确定哪些应用程序和服务在系统上使用得最多——opreport 能够用来确定一个应用程序或服务使用了多少处理器时间。如果系统运行了多个服务，但是运行状况不佳，消耗处理器时间最多的服务可以移动到专用系统上。
- 确定处理器使用情况——可以通过监视 CPU_CLK_UNHALTED 事件来确定在一个给定的时间周期内处理器的负载。该数据之后可以用来确定是否额外的处理器或者更快的处理器能够提高系统性能。

6.10.8. OProfile 对 Java 的支持

OProfile 允许您评测 Java 虚拟机（JVM）的动态编译代码（也被称作即时编译代码）。中标麒麟高级服务器操作系统软件 V7.0 中的 OProfile 包括了对 Java 虚拟机工具接口（JVM Tools Interface, JVMTI）代理库的内建支持，可以支持 Java1.5 或更高版本。

6.10.8.1. 评测 Java 代码

要使用 JVMTI 代理来评测 Java 虚拟机的即时编译代码，请添加以下 JVM 启动参数：

```
-agentlib:jvmti_oprofile
```

这里的 jvmti_oprofile 是 OProfile 代理的路径。对于 64 位的 JVM，路径如下所示：

```
-agentlib:/usr/lib64/oprofile/libjvmti_oprofile.so
```

当前，您可以添加--debug 命令行选项来启用调试模式。

要使用 OProfile 来评测即时编译代码，必须在系统中安装 oprofile-jit 软件包。有了这个包，您就能够显示方法级别的信息了。

取决于您所使用的 JVM，您可能需要为 JVM 安装 debuginfo 包。对于 Open JDK，debuginfo 包是需要安装的，而 Oracle JDK 则没有 debuginfo 包。为了让调试信息安装包和它们各自的非调试安装包保持同步，您还需要安装 yum-plugin-auto-update-debug-info 插件。这个插件会搜索调试信息仓库来寻找对应的更新。

成功配置之后，您就可以按照前面的各小节中描述的那样，使用标准的评测和分析工具了。

6.10.9. 图形界面

一些 OProfile 首选项可以通过图形界面来设置。请确保在您的系统上已经安装了提供 OProfile 图形用户界面的 oprofile-gui 软件包。要启动该界面，请在命令行提示符下以 root 用户执行 oprof_start 命令。

在对任何选项进行修改之后，请点击【Save and quit】按钮保存修改。首选项将被写入/root/.oprofile/daemonrc 文件中，然后应用程序退出。

退出应用程序不会让 OProfile 停止取样。

在【Setup】标签页，要像 6.10.3.2 设置要监视的事件中讨论的那样为处理器计数器设置事件，从下拉菜单中选择计数器，然后从列表中选择事件。一个简要的事件描述将会出现在列表下方的文本框中。列表中只会显示指定的计数器和指定的架构可用的事件。界面也会显示评测器是否正在运行，以及评测器的一些简要统计信息。

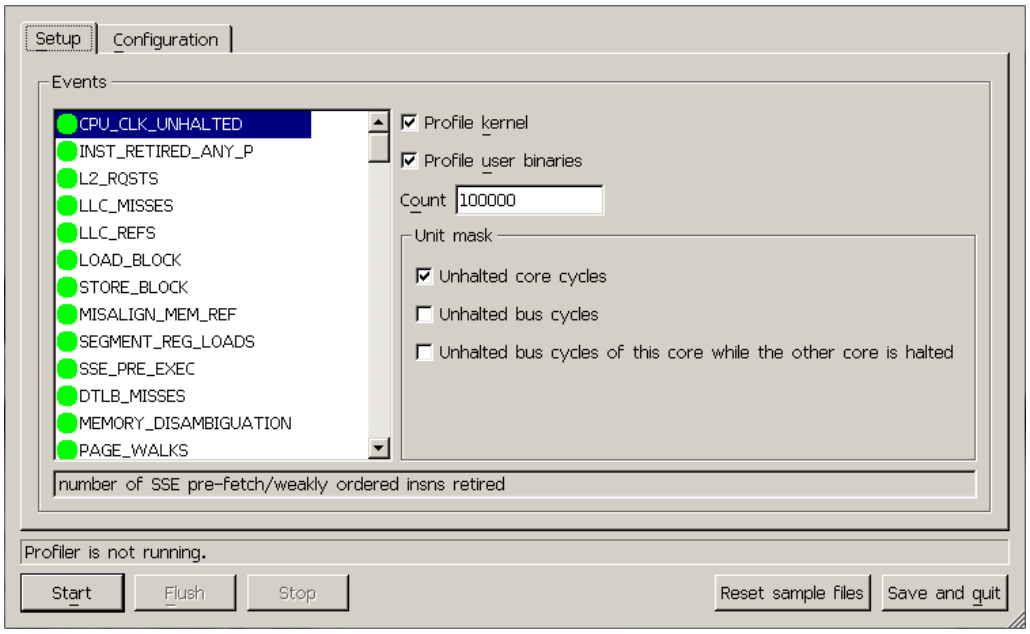


图 6-14 OProfile 设置

在标签页的右侧，选择【Profile kernel】选项来为当前选择的事件在内核模式下对事件进行计数，正如 6.10.3.3 分离内核和用户空间评测所讨论的那样。如果没有选择该选项，则不会为内核收集样本。

选择【Profile user binaries】选项来为当前选择的事件在用户模式下对事件进行计数，正如 6.10.3.3 分离内核和用户空间评测所讨论的那样。如果没有选择该选项，则不会为用户应用程序收集样本。

使用【Count】文本框来为当前选择的事件设置取样率，正如 6.10.3.2 中取样率所讨论的那样。

如果当前选择的事件有任何可用的单元掩码，它们将被显示在【Setup】标签页的右侧【Unit mask】区域中。选择单元掩码旁边的复选框来为事件启用单元掩码。

在【Configuration】标签页，要评测内核，请在【Kernel image file】文本框中为要监视的内核输入 vmlinux 文件的名称和位置。要配置 OProfile 不监视内核，请选择【No kernel image】。

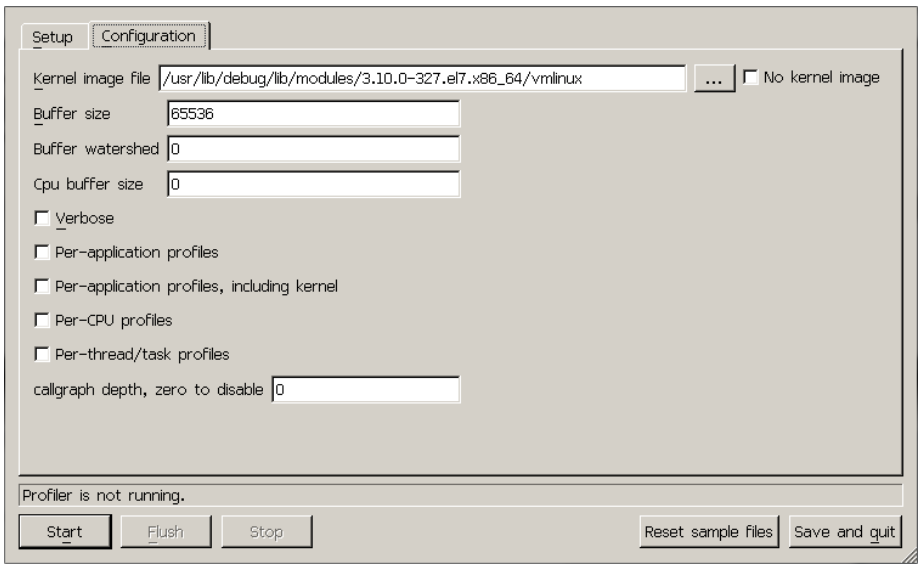


图 6-15 OProfile 配置

如果选择了【Verbose】选项，oprofiled 进程日志将包含更详细的信息。

如果选择了【Per-application profiles】，OProfile 将为库生成每个应用程序的评测。这相当于 opcontrol --separate=library 命令。如果选择了【Per-application profiles, including kernel】，OProfile 将为内核和内核模块生成每个应用程序的评测，正如 6.10.3.3 分离内核和用户空间评测所讨论的那样。这相当于 opcontrol --separate=kernel 命令。

要像 6.10.6 分析数据中讨论的那样，强制把数据写到样本文件中，请点击【Flush】按钮。这相当于 opcontrol --dump 命令。

要从图形界面启动 OProfile，请点击【Start】按钮。要停止评测器，请点击【Stop】按钮。退出应用程序不会让 OProfile 停止取样。

6.10.10. Oprofile 和 SystemTap

SystemTap 是一个追踪和探测工具，可以让用户比较详细地学习和监视操作系统的活动。它能够提供和 netstat、ps、top、iostat 等工具的输出类似的信息，不过，设计 SystemTap 是希望能够为收集到的信息提供更多的过滤和分析选项。

要了解为什么处理器在一个特定的代码区域花费时间以及具体在什么位置花费时间，而需要收集数据时，建议使用 OProfile。而要找出为什么处理器总是

处于空闲状态，就不太用得上 OProfile 了。

当您想在代码中的指定位置进行植入时，您可以使用 SystemTap。因为 SystemTap 可以让您进行代码植入，而不用停止或重启被植入的代码。它在向内核和守护进程中进行植入时特别有用。