KYLINSOFT
麒麟软件

银河麒麟高级服务器操作系统 V10

系统管理员手册

麒麟软件有限公司

2024 年 03 月

目　录

麒麟软件有限公司（简称“麒麟软件”）

为顺应产业发展趋势、满足国家战略需求、保障国家网络空间安全、发挥中央企业在国家关键信息基础设施建设中主力军作用，中国电子信息产业集团有限公司（简称“中国电子”）于 2019 年 12 月将旗下天津麒麟信息技术有限公司和中标软件有限公司强强整合，成立麒麟软件有限公司（简称“麒麟软件”），打造中国操作系统核心力量。

麒麟软件主要面向通用和专用领域打造安全创新操作系统产品和相应解决方案，以安全可信操作系统技术为核心，现已形成银河麒麟服务器操作系统、桌面操作系统、嵌入式操作系统、麒麟云、操作系统增值产品为代表的产品线。麒麟操作系统能全面支持飞腾、鲲鹏、龙芯等六款主流国产 CPU，在安全性、稳定性、易用性和系统整体性能等方面远超国内同类产品，实现国产操作系统的跨越式发展。目前，公司旗下产品已全面应用于党政、金融、交通、通信、能源、教育等重点行业，服务用户覆盖所有的中央部委、政府机关、地市党委。根据赛迪顾问统计，麒麟软件旗下操作系统产品，连续 12 年位列中国 Linux 市场占有率第一名。

麒麟软件注重核心技术创新，2018 年荣获“国家科技进步一等奖”，2020 年发布的银河麒麟操作系统 V10 被国资委评为“2020 年度央企十大国之重器”，相关新闻入选中央广播电视总台“2020 年度国内十大科技新闻”。麒麟软件荣获“中国电力科学技术进步奖一等奖”、“中国品牌日电子信息行业国货新品”等国家级、省部级和行业奖项 400 余个，并被授予“国家规划布局内重点软件企业”、“国家高技术产业化示范工程”、“科改示范行动企业”、“国有重点

企业管理标杆创建行动标杆企业”等称号。通过 CMMI5 级评估，现有博士后工作站、省部级企业技术中心、省部级基础软件工程中心等，先后申请专利 551 项，其中授权专利 214 项，登记软件著作权 553 项，主持和参与起草国家、行业、联盟技术标准 60 余项。

麒麟软件在北京、天津、上海、长沙、广州、深圳、太原、郑州、武汉、南京、南昌、济南、南宁、成都、沈阳、厦门等地设有分支机构，服务网点遍布全国 31 个省会城市和 2 个计划单列市。

麒麟软件高度重视生态体系建设，与众多软硬件厂商、集成商建立长期合作伙伴关系，建设完整的自主创新生态链，为国家网信领域安全创新提供有利支撑。截止 2024 年 1 月 16 日，麒麟软件已与 19300 多家厂商建立合作，完成超 422 万项软硬件认证和适配，生态适配官网累计注册用户数超 5.9 万+人。

麒麟软件积极贯彻人才是第一资源的理念，以麒麟软件教育发展中心为组织平台，联合政产学研各方力量，探索中国特色的网信人才培养模式，目前已形成了源自麒麟软件核心技术的“5 序”培训认证体系、课件体系、教材体系、师资体系、平台体系，并与工信部教育与考试中心联合推出“百城百万”操作系统培训专项行动，持续为我国培养各类操作系统专业人才。

在开源建设方面，麒麟软件正式发布中国首个桌面操作系统根社区 openKylin，通过构建自有可靠的开源软件供应链，具备自主可持续发展能力，持续贡献主流上游开源项目，携手十余家产业同仁共建 openKylin，力争成为具有国际影响力的顶级开源社区。此外，麒麟软件在 OpenStack 社区贡献位列国内第一、全球第三；作为 openEuler 开源社区发起者，以 Maintainer 身份承

担 80 个项目，除华为公司外贡献第一；主导开发优麒麟开源操作系统，全球累计下载量数千万次，活跃爱好者和开发者数十万人。

银河麒麟最终用户使用许可协议

尊敬的银河麒麟操作系统及相关产品用户（以下称“您”或“贵机构”）：

首先感谢您选用由麒麟软件有限公司开发并制作发行的银河麒麟操作系统软件产品。

请在打开本软件介质包之前，仔细阅读本协议条款、提供的所有补充许可条款（统称“协议”）及银河麒麟操作系统隐私政策声明。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，麒麟软件有限公司（下称“麒麟软件”）向您授予非排他、不可转让的许可，仅允许内部使用由麒麟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，麒麟软件给予贵机构非独占、不可转让、有限的许可，允许贵机构至多使用软件的五(5)份完整及未经修改的二进制格式副本，而此种软件副本仅可安装于贵机构操作的电脑中。

– 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，麒麟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的文件。

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。麒麟软件和/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。对于麒麟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。

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、修改及分发银河麒麟操作系统软件产品中遵守《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》协议的银河麒麟操作系统软件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，但任何以银河麒麟操作系统软件产品为基础的衍生发行版未经麒麟软件有限公司的书面授权不能使用任何麒麟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，如银河麒麟操作系统软件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。

3. 有限担保

麒麟软件向您担保，自购买或其它合法取得之日起九十（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，您的所有补偿及麒麟软件的全部责任为由麒麟软件选择更换本软件介质或退还本软件的购买费用。

4. 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的条件、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的担保，均不予负责，但上述免责声明被认定为法律上无效的情况除外。

5. 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的或与之相关的任何收益损失、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，麒麟软件或其许可方均不承担任何责任（即使麒麟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，麒麟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，上文所述的限制仍然适用。

6. 终止

本协议在终止之前有效。您可以随时终止本协议，但必须同时销毁本软件的

全部正本和副本。如果您未遵守本协议的任何规定，则本协议将不经麒麟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，并且需承担因未遵守本协议而导致的法律责任。

7.法律适用

与本协议相关的任何争议解决（包括但不限于诉讼、仲裁等）均适用中华人民共和国法律。任何其它国家和地区的法律规则不予适用。

8. 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，除非该删除会防碍各方根本目的的实现（在这种情况下，本协议将立即终止）。

9. 完整性

本协议是您与麒麟软件就其标的达成的完整协议。它取代此前或同期的所有和本协议不一致的口头或书面往来信息、建议、陈述和担保，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。

10. 商标和标识

贵机构承认并与麒麟软件有着以下共识，即麒麟软件拥有麒麟软件、银河麒麟商标，以及所有与麒麟软件、银河麒麟相关的商标、服务标记、标识及其他品牌标识（“麒麟软件标记”）。贵机构对麒麟软件标记的任何使用都应有利于麒麟软件。

11. 源代码

本软件可能包含源代码，其提供之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。

12. 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。

产品中本协议提供中英文两种版本，以上任何内容如有歧义，以中文版本为准。

银河麒麟操作系统隐私政策声明

版本发布日期：2019 年 12 月 18 日

版本生效日期：2019 年 12 月 18 日

尊敬的银河麒麟操作系统用户（以下简称“您”），银河麒麟操作系统系列软件产品是由麒麟软件有限公司（以下简称“我们”或“麒麟软件”）研制发行的，用于办公或构建企业及政府的信息化基础设施。

麒麟软件非常重视您的个人信息和隐私保护，在您使用本产品的过程中，我们会按照《银河麒麟操作系统隐私政策声明》（以下简称“本声明”）收集、存储、使用您的个人信息。为了保证对您的个人隐私信息合法、合理、适度的收集、使用，并在安全、可控的情况下进行传输、存储，我们制定了本声明。我们将向您说明收集、保存和使用您的个人信息的方式，以及您访问、更正、删除和保护这些信息的方式。我们将会按照法律要求和业界成熟安全标准，为您的个人信息提供相应的安全保护措施。如您点击或勾选“同意”并确认提交，即视为您同意本隐私政策声明，并同意我公司将按照本政策来收集、存储和使用您的相关信息。

本声明将帮助您了解以下内容：

一、关于收集和使用涉及您的个人信息

二、如何存储和保护涉及您的个人信息

三、如何管理您的个人信息

四、关于第三方软件的隐私说明

五、关于未成年人使用产品

六、本声明如何更新

七、如何联系我们

一、如何收集和使用您的个人信息

1.收集涉及您的个人信息的情况

我们在您使用银河麒麟操作系统产品过程中收集相关的信息，主要为了向您提供更高质量、更易用的产品和更好的服务。

1）银河麒麟操作系统的产品授权许可机制，会根据您所使用计算机的网卡、固件和主板等信息通过加密机制和转换方法生成申请产品正式授权许可的机器码；您将该机器码发给麒麟软件商务人员根据合同及相关协议可申请正式许可。该机器码不包含您所使用计算机的网卡、固件和主板等设备具体信息。

2）银河麒麟操作系统应用商店的服务器端，会根据您所使用计算机的 CPU 类型信息以及 IP 地址进行连接；实现您方便快捷使用应用商店。您所使用计算机的 IP 地址可能会记录在应用商店的服务器端系统的日志中。

3）银河麒麟操作系统的升级更新，会根据您所使用计算机的 IP 地址进行连接；以便实现您确认是否更新升级系统。

4）使用银河麒麟操作系统产品过程中，因业务往来及技术服务等您提供的电子邮箱、电话、姓名等个人信息。

5）银河麒麟操作系统可能提供生物识别相关功能，会存储身份鉴别相关的信息在您的机器。这部分信息我们不收集和上传服务器。

以后银河麒麟操作系统产品升级过程中，如新增涉及个人信息收集部分，将及时更新本部分内容。

2. 使用涉及您的个人信息的情况

我们严格遵守法律法规的规定及与用户的约定,将收集的信息用于以下用途。若我们超出以下用途使用您的信息,我们将再次向您进行说明,并征得您的同意。我们会将收集的信息用于以下用途:

产品功能:主要涉及产品许可机制、应用商店使用、系统更新维护、生物识别等需要。

安全保障:为保障您使用银河麒麟操作系统的安全,我们会利用相关信息协助提升产品的安全性、可靠性和可持续服务。

与您沟通:我们会利用收集的信息(例如您提供的电子邮件地址、电话等)直接与您沟通。例如,业务联系、技术支持或服务回访。

产品改进:将收集的信息用于改进产品当前的易用性、缺陷以及提升产品用户体验等。

为了遵从相关法律法规、部门规章、政府指令的相关要求。

3.信息的分享及对外提供

我们不会共享或转让您的个人信息至第三方,但以下情况除外:

1)获取您的明确同意:经您事先同意,我们可能与第三方分享您的个人信息;

2)为实现外部处理的目的,我们可能会与关联公司或其他第三方合作伙伴(第三方服务供应商、承包商、代理、应用开发者等)分享您的个人信息,让他们按照我们的说明、隐私政策以及其他相关的保密和安全措施来为我们处理上述信息,并用于向您提供我们的服务,实现"如何收集和使用您的个人信息"部分所

述目的。如我们与上述关联公司或第三方分享您的信息，我们将会采用加密、匿名化处理等手段保障您的信息安全。

3）我们不会对外公开披露所收集的个人信息，如必须公开披露时，我们会向您告知此次公开披露的目的、披露信息的类型及可能涉及的敏感信息，并征得您的明示同意。

4）随着我们业务的持续发展，我们有可能进行合并、收购、资产转让等交易，我们将告知相关情形，按照法律法规及不低于本声明所要求的标准继续保护或要求新的控制者继续保护您的个人信息。

5）我们可能基于法律要求或相关部门的执法要求披露您的个人信息。

如我们使用您的个人信息，超出了与收集时所声称的目的及具有直接或合理关联的范围，我们将在使用您的个人信息前，再次向您告知并征得您的明示同意。

根据相关法律法规以及国家标准，在以下情况下我们可能会收集、使用您的个人信息，征得授权同意的例外情况：

1）与国家安全、国防安全等国家利益直接相关的；

2）与公共安全、公共卫生、公众知情等重大公共利益直接相关的；

3）与犯罪侦查、起诉、审判和判决执行等直接相关的；

4）出于维护您或其他个人的生命、财产等重大合法权益但又很难得到您本人同意的；

5）所收集的个人信息是您自行向社会公众公开的；

6）从合法公开披露的信息中收集的个人信息，如合法的新闻报道、政府信息公开等渠道；

7）根据您要求签订和履行合同所必需的；

8）用于维护所提供的产品或服务的安全稳定运行所必需的。如发现、处置产品或服务的故障；

9）出于公共利益开展统计或学术研究所必需，且其对外提供学术研究或描述的结果时，对结果中所包含的个人信息进行去标识化处理的；

10）法律法规规定的其他情形。

二、我们如何存储和保护涉及您的个人信息

1.信息存储的地点

我们会按照法律法规规定，将在中国境内收集和产生的个人信息存储于中国境内。

2.信息存储的期限

一般而言，我们仅为实现目的所必需的时间保留您的个人信息。记录在日志中的信息会按配置在一定期限保存及自动删除。

当我们的产品或服务发生停止运营的情形时，我们将以通知、公告等形式通知您，在合理的期限内删除您的个人信息或进行匿名化处理，并立即停止收集个人信息的活动。

3.我们如何保护这些信息

我们努力为用户的信息安全提供保障，以防止信息的丢失、不当使用、未经授权访问或披露。

我们将在合理的安全水平内使用各种安全保护措施以保障信息的安全。例如，我们会使用加密技术（例如，SSL/TLS）、匿名化处理等手段来保护您的个人信

息。

我们建立专门的管理制度、流程和组织以保障信息的安全。例如，我们严格限制访问信息的人员范围，要求他们遵守保密义务，并进行审计。

4.若发生个人信息泄露等安全事件，我们会依法启动应急预案，阻止安全事件扩大，并以推送通知、公告等形式告知您安全事件的情况、事件可能对您的影响以及我们将采取的补救措施。我们还将按照法律法规和监管部门要求，上报个人信息安全事件的处置情况。

三、如何管理您的个人信息

如果担心因使用银河麒麟操作系统产品导致个人信息的泄露，您可根据个人及业务需要考虑暂停或不使用涉及个人信息的相关功能，如产品正式授权许可、应用商店、系统更新升级、生物识别等。

在使用银河麒麟操作系统之上使用第三方软件时，请注意个人隐私保护。

四、关于第三方软件的隐私说明

您在使用银河麒麟操作系统之上安装或使用第三方软件时，第三方软件的隐私保护和法律责任由第三方软件自行负责。

您在使用银河麒麟操作系统之上安装或使用第三方软件时，请您仔细阅读和审查对应的隐私声明或条款；注意个人隐私保护。

五、关于未成年人使用产品

银河麒麟操作系统系列产品仅供成年人使用，如果您是未成年人，则需要您的监护人同意您使用本产品并同意相关服务条款。父母和监护人也应采取适当的预防措施保护未成年人，包括监督其对银河麒麟操作系统系列产品的使用。

六、本声明如何更新

我们保留适时更新本声明的权利，当本声明发生变更时，我们会通过产品安装过程或公司网站向您展示变更后的声明，只有在获取您的同意后，我们才会按照更新后的声明收集、使用、存储您的个人信息。

七、如何联系我们

如您对本声明存在任何疑问，或任何相关的投诉、意见，请联系麒麟软件客服热线 400-089-1870、官方网站（www.kylinos.cn）以及麒麟软件进行咨询或反映。您可以通过发送邮件至 market@kylinos.cn 方式与我们联系。

受理您的问题后，我们会及时、妥善处理。一般情况下，我公司将在 15 个工作日内给予答复。

本声明自更新之日起生效，同时提供中英文两种版本，以上任何条款如有歧义，以中文版本为准。

最近更新日期：2019 年 12 月 18 日　　麒麟软件有限公司

地址：天津市滨海高新区塘沽海洋科技园信安创业广场 3 号楼（300450）

北京市海淀区北四环西路 9 号银谷大厦（100190）

长沙市开福区三一大道 156 号工美大厦（410073）

上海市徐汇区番禺路 1028 号数娱大厦（200030）

电话：天津（022）58955650　北京（010）51659955　长沙（0731）88280170 上海（021）51098866

传真：天津（022）58955651　北京（010）62800607 长沙（0731）88280166

上海（021）51062866

公司网站：www.kylinos.cn

电子邮件：support@kylinos.cn

特别提示说明

银河麒麟高级服务器操作系统 V10 同源支持飞腾、龙芯、申威、兆芯、海光、鲲鹏等自主 CPU 平台。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，有需要请致电麒麟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，请以银河麒麟高级服务器操作系统 V10 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/locale.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 1-1 在所示。

表 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=locale
```

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

显示可用键盘布局列表可使用如下命令：

```
#localectl list-keymaps
```

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},如“cz”，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。同样也可用以下命令单独配置 X11 窗口的键盘布局：

```
#localectl set-x11-keymap cz
```

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 cz
```

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 1.6 获取特权章节和 5.1 章节。

1.2. 网络访问配置

1.2.1. 动态网络配置

打开终端，以网口 eth0 为例：

```
#nmcli conn add connection.id eth0-dhcp type ether ifname eth0 ipv4.method auto
```

其中“eth0-dhcp”为连接的名字，可以根据自己的需要命名方便记忆和操作的名字；“ifname eth0”为配置的网口，根据自己的设备情况按需调整。

1.2.2. 静态网络配置

打开终端，以网口 eth0 为例：

```
#nmcli conn add connection.id eth0-static type ether ifname eth0 ipv4.method manual ipv4.address 192.168.1.10/24 ipv4.gateway 192.168.1.254 ipv4.dns 192.168.1.254
```

其中“eth0-static”为连接的名字，可以根据自己的需要命名方便记忆和操作的名字；“ifname eth0”为配置的网口，根据设备情况按需调整；IP、子网掩码、网关根据实际网络按需配置。

1.2.3. 配置 DNS

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

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

1.3. 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。

1.3.1. timedatectl 工具使用说明

1.3.1.1. 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 systemd-timedated.service
```

1.3.1.2. 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。

1.3.1.3. 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'
```

1.3.1.4. 修改时区

执行以下命令可以显示当前时区：

```
#timedatectl show
```

以 root 用户运行以下命令可以修改当前时区，如修改为“上海”：

```
#timedatectl set-timezone Asia/Shanghai
```

显示所有时区命令如下：

```
#timedatectl list-timezones
```

1.3.1.5. 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，可以执行以下命令：

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

1.3.2. date 工具使用说明

1.3.2.1. 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 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
```

1.3.2.2. 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
```

1.3.2.3. 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
```

1.3.3. hwclock 工具使用说明

1.3.3.1. 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
```

1.3.3.2. 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
```

1.3.3.3. 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
```

1.4. 用户配置

用户配置可以对用户进行创建与管理，点击**开始->控制面板->用户账户->创建一个新账户**可以新建用户，并且可以配置头像、是否自动登录以及用户类型，输入用户名以及密码，点击创建用户即可创建成功。

图 1-1 创建账户

用户创建成功后，可以对账户密码以及头像进行重新更改，点击“更换密码”以及“更换头像”即可更改，点击“删除用户”即可将用户删除。

图 1-2 用户账户

1.5. Kdump 机制

Kdump 是基于 kexec 的内核崩溃转储机制，在系统崩溃、死锁或死机时用来转储内存运行参数的一个工具和服务，用来捕获内核崩溃的时候产生的 crash dump。Kdump 是迄今为止最可靠的内核转存机制，最大的优点在于崩溃转储数据可从一个新启动内核的上下文中获取，而不是从已经崩溃内核的上下文。

Kdump 需要两个不同目的的内核，生产内核和捕获内核。生产内核是捕获内核服务的对像：如果系统一旦崩溃，那么正常的内核就没有办法工作了，在这个时候将由 Kdump 产生一个用于捕获当前运行信息的内核，该内核会与相应的 ramdisk（虚拟内存盘：将内存模拟成硬盘的技术）一起组建一个微环境，将此

时的内存中的所有运行状态和数据信息收集到一个 dump core 文件中，一旦内存信息收集完成，系统将会自动重启。

Kdump 机制主要包括两个组件：kdump 和 kexec。

kdump 使用 kexec 启动到捕获内核，以很小内存启动以捕获转储镜像。生产内核保留了内存的一部分给捕获内核启动用。由于 kdump 利用 kexec 启动捕获内核，绕过了 BIOS，所以第一个内核的内存得以保留。这是内核崩溃转储的本质。

kexec 是一个快速启动 kernel 的机制，它运行在某一正在运行的 kernel 中，启动一个新的 kernel 而且不用重新经过 BIOS 就可以完成启动。因为一般 BIOS 都会花费很长的时间,尤其是在大型并且同时连接许多外部设备的 Server 上的环境下，BIOS 会花费更多的时间。

kexec 包括 2 个组成部分：一是内核空间的系统调用 kexec_load，负责在生产内核启动时将捕获内核加载到指定地址。二是用户空间的工具 kexec-tools，他将捕获内核的地址传递给生产内核,从而在系统崩溃的时候能够找到捕获内核的地址并运行。

1.5.1. Kdump 命令行配置

1.5.2.1.安装 Kdump 需要的软件包

表 1-6 Kdump 所需的软件包

软件包名称	软件包说明
kdump	kdump软件包
kexec-tools	kexec软件包，kdump用到的各种工具都在此包中

kernel-debuginfo	用来分析vmcore文件
crash	vmcore分析工具

使用Kdump服务，需先安装这些工具包。安装命令如下：

```
dnf install kdump kexec-tools kernel-debuginfo-common kernel-debuginfo
```

1.5.2.2.配置 grub

Kdump 的使用需要配置 kdump kernel 的内存区域。Kdump 要求操作系统正常使用的时候，不能使用 kdump kernel 所占用的内存，配置这个需要修改/boot/grub/grub.conf 文件，修改用到的引导部分，加入 crashkernel。Crashkernel 的格式如下：

```
crashkernel=nn[KMG]@ss[KMG]
```

其中 nn 表示要为 crashkernel 预留多少内存，ss 表示为 crashkernel 预留内存的起始位置。

修改完成并重启后，可以通过 cat /proc/cmdline 查看 kernel 启动配置选项，其中已经加入了 crashkernel 项。

1.5.2.3.启动和查看 Kdump 服务

查看 Kdump 服务命令如下：

```
#systemctl status kdump.service
```

如果 Kdump 服务未开启，使用如下命令来启动 kdump 服务:

```
#systemctl start kdump.service
```

1.6. 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。

1.6.1. 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
```

当将用户加入 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。

1.6.2. 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
```

第二章 基本开发环境

2.1. Qt-5.14.2

Qt 是一个跨平台的桌面、嵌入式和移动应用程序开发框架。只需重新编译即可将现有的桌面或嵌入式应用程序带到移动设备中。有很强的图形功能和性能。Qt5 是 Qt 的最新版本，开发人员能够以直观的用户界面针对多个目标开发应用程序。通过 Qt5 中改进的 JavaScript 和 QML 支持，开发人员可以更加高效和灵活，同时仍支持 C++和 Qt Widget。Qt5 与 Qt4 高度兼容，并借助模块化的代码库和 Qt Platform Abstraction，增强了代码的可移植性。

当前更新至版本 5.14.2，更多新特性请查阅

https://doc.qt.io/archives/qt-5.14

2.2. GCC-7.3

GCC 是由 GNU 开发的编程语言编译器，支持多种语言的编译，例如 C，C++，Objective-C，Java 等，同时包含这些语言的库文件。GCC 是一种开源的开发工具，支持多种体系架构，易于扩展和测试。

主要特性包括：

•支持 GNU 标准；

•编译器基于 GPL 标准；

•具有不断更新的运行时库，调试效率高；

详细用法或其他资料请查阅 https://gcc.gnu.org/。

2.3. GDB-9.2

GDB 是 GNU 代码调试器，允许查看程序内部执行流程，或者程序在发生异常时的状态。GDB 的功能主要包括：

执行一些能够影响程序运行结果的操作；

在指定的条件下停止程序；

在程序停止运行时，检查此时程序内部发生了什么；

修改程序，以验证程序 bug 对程序的影响，同时了解到程序中许多其他的内容，例如变量取值等。

GDB 支持以下编程语言：

Ada
Assembly
C
C++
D
Fortran
Go
Objective-C
OpenCL
Modula-2
Pascal
Rust

终端控制台输入命令”gdb”，即可打开 GDB 调试工具。

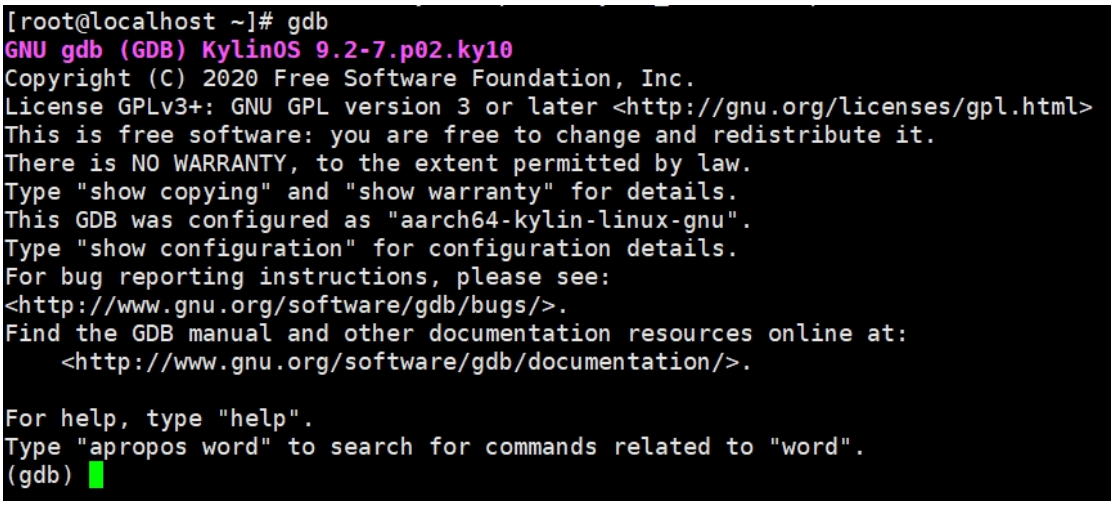


图 2-1 显示页面

当前更新至版本 9.2，详细用法或其他资料请查阅

http://gnu.org/software/gdb/。

2.4. Python3-3.7.9

Python 是一种清晰而强大的面向对象编程语言，可与 Perl、Ruby、Scheme 或 Java 相媲美。

Python 的一些显著特性:

语法简洁，程序易于阅读。

程序运行简单，这使得 Python 成为许多编程任务的理想选择，同时又不影响可维护性。

附带一个大型标准库，支持许多常见的编程任务，如连接网络服务器、用正则表达式搜索文本、读取和修改文件等。

Python 的交互模式使得测试简短的代码片段变得很容易，python 开发环境叫做 IDLE。

通过添加以编译语言(如 C 或 C++)实现的新模块,可以轻松地进行扩展。

也可以嵌入到应用程序中以提供可编程接口。

在任何地方运行，包括 Mac OS X，Windows，Linux 和 Unix，非官方版本也可用于 Android 和 iOS。

Python 是一款免费软件。下载或使用 Python，或者将它包含在您的应用程序中不需要任何费用，Python 也可以被自由修改和重新发布。

Python 的一些编程语言特性包括:

有多种基本数据类型可用:number(floating point,complex 和 unlimited-length long integers)、strings(包括 ASCII 和 Unicode)、lists 和 dictionaries。

Python 支持带有类和多重继承的面向对象编程。

代码可以分成模块和包。

该语言支持引发和捕获异常，从而实现更清晰的错误处理。

数据类型是强类型和动态类型。混合不兼容的类型(例如，试图添加一个字符串和一个数字)会导致引发异常，从而更快地发现错误。

Python 包含高级编程特性，如 generators 和 list comprehensions。

Python 的自动内存管理使您不必手动分配和释放代码中的内存。

当前更新至版本 3.7.9，详细资料请查阅 https://www.python.org/。

2.5. Openjdk-1.8.0

Opengdk 作为 GPL 许可的 Java 平台开源化实现，由 Sun 公司开发，提供了一个 java 的运行环境，支持 Solaris,Linux,Mac OS X 或 Windows 多种操作系统。

新版本的新特性主要有：

Lambda 表达式和 Stream API；

时间与日期 API；

构造器引用；

红黑树的使用使运行速度更快；

减少空指针异常等。

当前更新至版本 1.8.0，详细资料请查阅 http://openjdk.java.net。

第三章 常用图形化工具

3.1. 刻录工具

光盘刻录器是一款便捷的刻录工具，点击**开始->所有程序->系统工具->光盘刻录器**，可以创建音频光盘、数据 CD/DVD、视频 DVD/SVCD 以及镜像的刻录，如下图：

图 3-1 刻录工具

3.2. 磁盘

3.2.1. 磁盘管理

3.2.1.1. 磁盘管理工具介绍

磁盘管理工具是一款能够查看并管理磁盘分区以及创建和恢复分区的工具，可以用它进行创建和格式化分区、挂载和卸载卷组以及其它相关的磁盘操作。

3.2.1.2. 磁盘管理工具界面展示

点击左下角-【开始】，界面会弹出菜单栏，依次选择**所有程序->附件->磁盘**。

图 3-2 显示应用程序

磁盘管理界面，左侧显示目前现有设备和支持设备目录，右侧显示选中设备的详细使用情况。能够识别到外部硬盘和额外的硬盘驱动器。

图 3-3 磁盘管理界面

3.2.2. 磁盘管理工具使用

3.2.2.1. 分区创建

选择剩余空间，点击【+】图标弹出创建分区页面，拖动按钮选择创建分区的大小，点击下一步，输入卷名和选择类型后点击【创建】，创建完成后图形界面显示已创建分区。

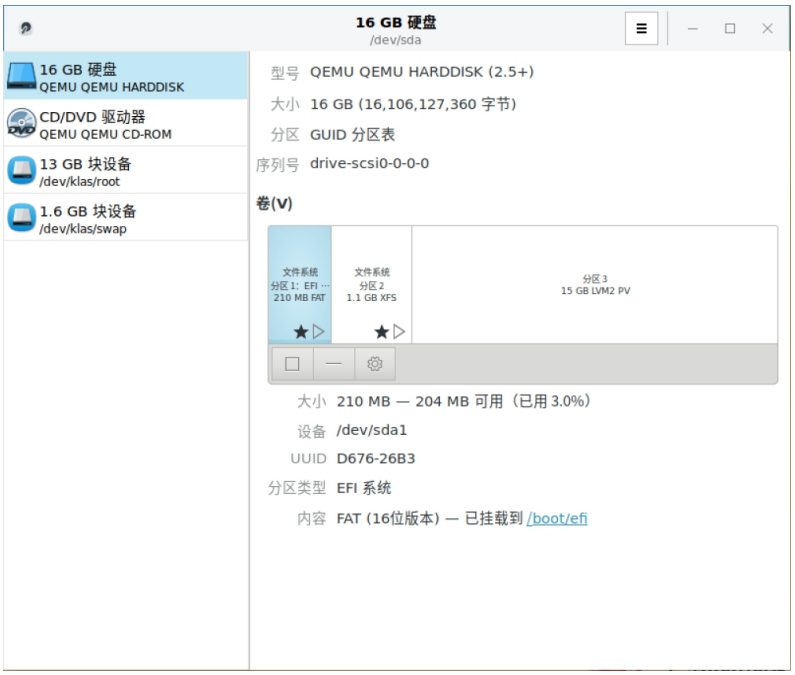


图 3-4 创建分区

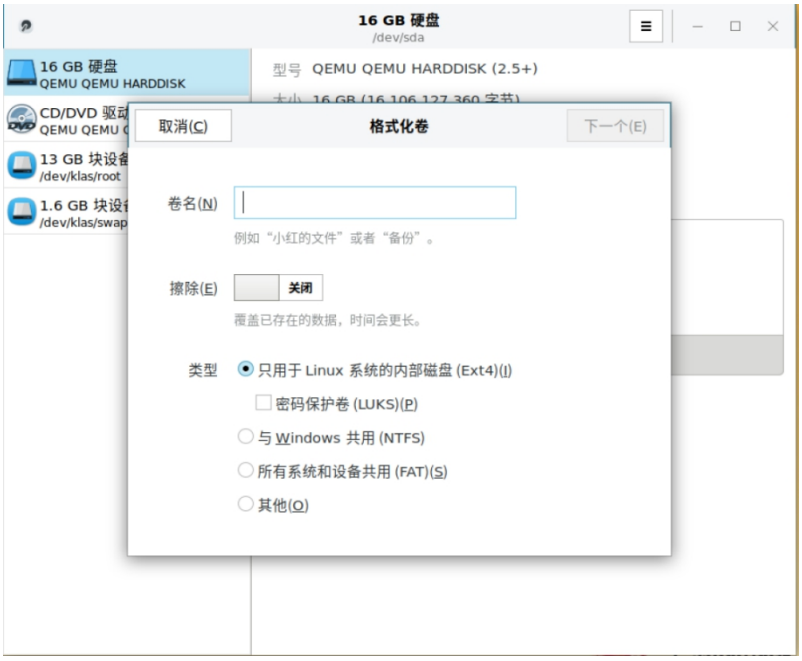


图 3-5 格式化卷

3.2.2.2. 分区格式化

选择需要格式化的磁盘或分区，选中后点击设置弹出下拉菜单，选择格式化分区，输入卷名，选择类型后，点击下一步，点击【格式化】，磁盘开始格式化，格式化成功后即可正常使用。

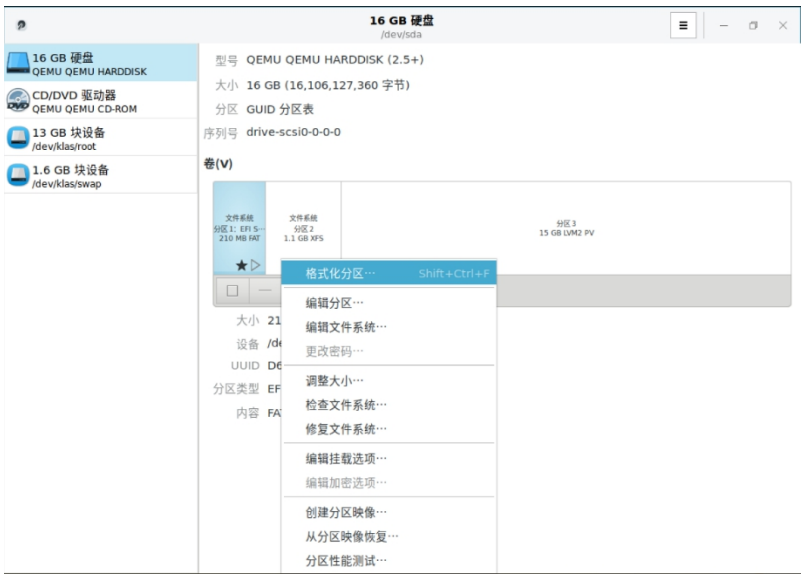


图 3-6 格式化

3.2.2.3. 分区编辑

选择需要编辑的分区，点击下方【设置】按钮弹出下拉菜单，点击【编辑分区】，在弹出的编辑分区窗口中选择对应的类型，点击更改即可。

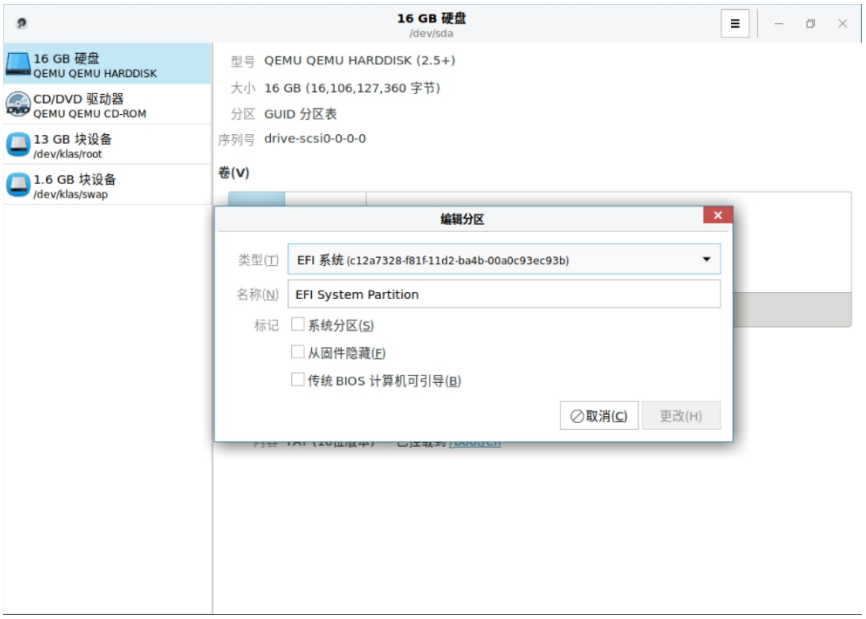


图 3-7 编辑分区

3.2.2.4. 编辑文件系统

选择需要修改名称的分区，点击【设置】按钮，弹出下拉菜单，点击【编辑文件系统】，弹出编辑文件系统的窗口，输入新的分区名称，点击更改即可。

图 3-8 编辑文件系统

3.2.2.5. 分区大小调整

选择需要修改大小的分区，点击【设置】按钮，弹出下拉菜单，点击【调整大小】，弹出调整大小窗口，拖动选择大小或者手动输入大小，点击调整大小即可。

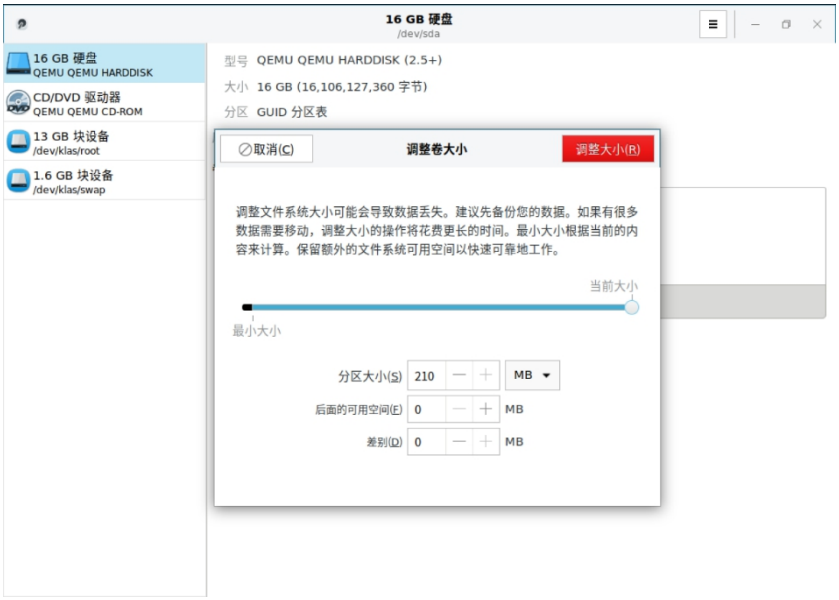


图 3-9 编辑调整大小

3.2.2.6. 分区卸载和挂载

选择需要卸载的分区，点击下侧的【□】图案进行卸载，

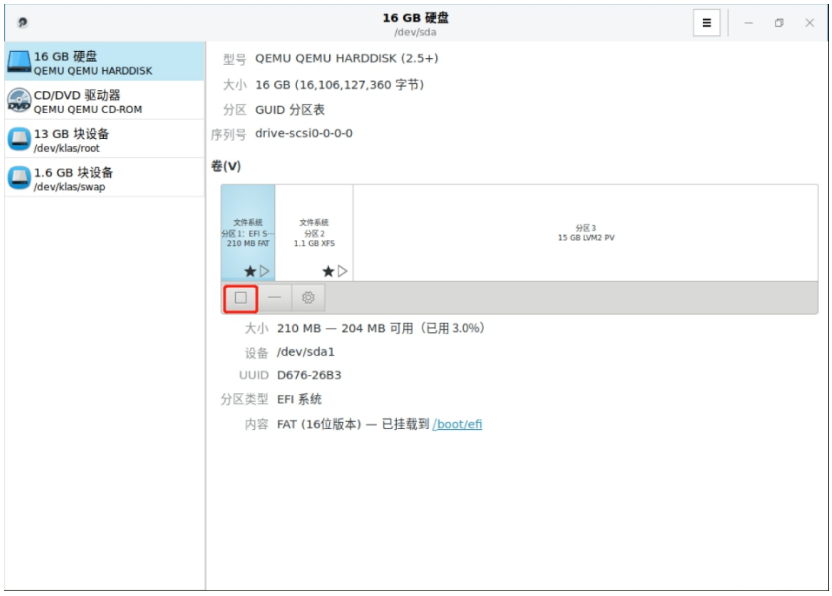


图 3-10 分区卸载

卸载完成后，内容显示为未挂载，此时卸载时的□变成了▷

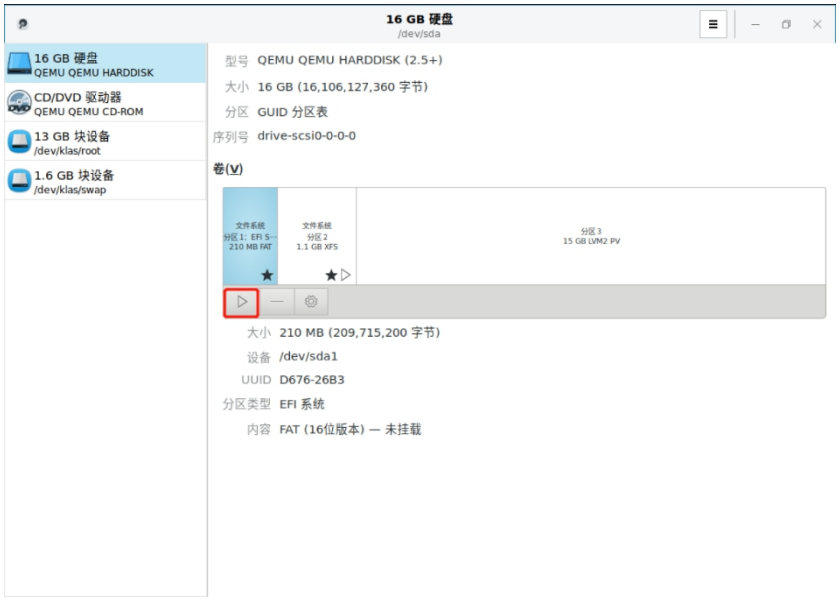


图 3-11 分区卸载完成

点击【▷】进行挂载，挂载完成后，内容显示为已挂载到对应目录。

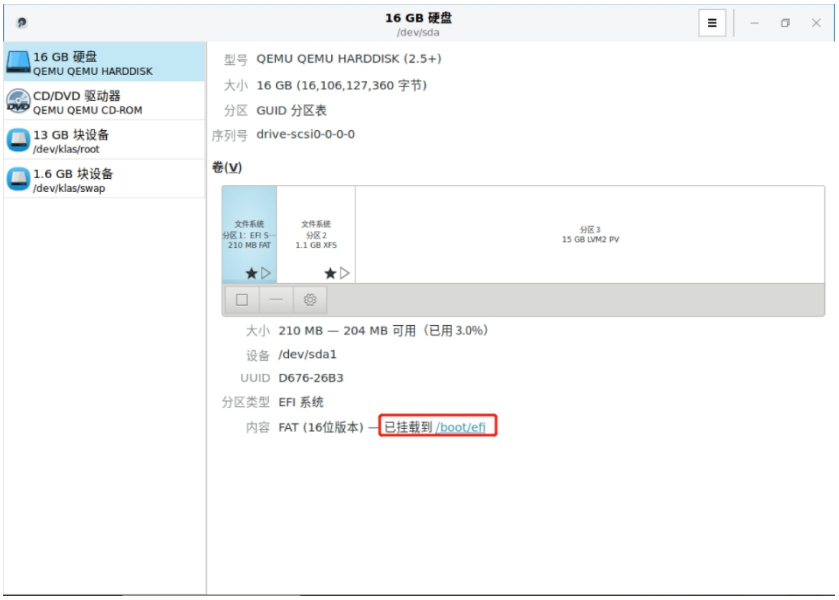


图 3-12 分区挂载

3.2.2.7. 分区删除

选择要删除的分区，点击【-】弹出删除分区的提示，点击【删除】即可。

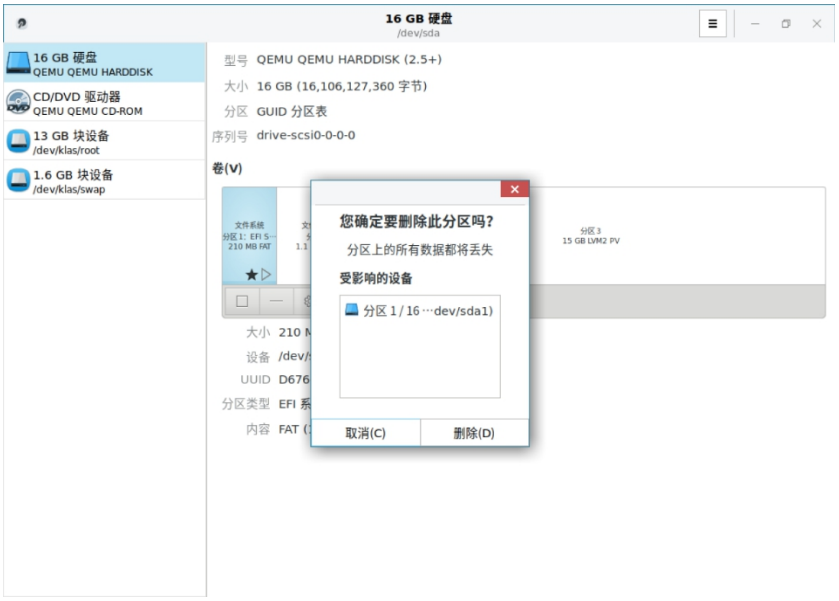


图 3-13 分区删除

3.3. 远程桌面

3.3.1. VNC 查看器

TigerVNC Viewer 是一款方便的远程桌面查看端，点击**开始->所有程序->**

互联网->TigerVNC 查看器，输入 VNC 服务器信息点击连接，即可连接。

图 3-14 VNC 查看器

点击“选项”按钮，可以对压缩、安全、输入、屏幕以及杂项等进行相关配置。

图 3-15 VNC 查看器配置

3.3.2. 远程查看程序 SSH

ssh 工具可以让用户登录到一台远程机器上，并在该机器上执行命令。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
```

该命令将会以用户正在使用的本地机器的用户名登录。如果用户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
```

用户第一次连接时，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 SHA256:Ixy64icRYc/h7XSOvUVywS7t7ThtmOsPT1s07wDD5P8.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

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 SHA256:b0gGbI+Xk/l+Ve76j3mqpYSty0n3gR9QiIsd+8oV3GIno comment (ECDSA)
```

输入 yes 接受密钥并确认连接。用户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，验证服务端没有受到攻击。

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 #Host penguin.example.com found: line 15 type ECDSA /home/USER/.ssh/known_hosts updated. Original contents retained as /home/USER/.ssh/known_hos ts.old
```

在输入密码之后，用户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 release V10 (Lance)
```

在用户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

用户将返回到本地的 shell 命令行提示符下。

3.4. Cockpit 远程管理

3.4.1. Cockpit

Cockpit 是一个 Web 控制台，具有易于使用的基于 Web 的界面，使您可以在服务器上执行管理任务。

Cockpit Web 控制台使您可以执行多种管理任务，包括：管理服务，管理用户帐号，管理和监视系统服务，配置网络接口和防火墙，查看系统日志，管理虚拟机，创建诊断报告，设置内核转储配置，配置 SELinux，更新软件，管理系统订阅等。

Cockpit Web 控制台使用与终端相同的系统 API，并且在终端中执行的任务会迅速反映在 Web 控制台中。此外，您还可以直接在 Web 控制台中或通过终端配置设置。

3.4.2. 启动和查看 Cockpit 服务

使用如下命令启动和查看 cockpit 服务：

```
#systemctl enable --now cockpit.socket
```

```
[root@localhost ~]# systemctl enable --now cockpit.socket
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/system/coc
kpit.socket.
```

```
#systemctl status cockpit.socket
```

```
[root@localhost ~]# systemctl status cockpit.socket
● cockpit.socket - Cockpit Web Service Socket
   Loaded: loaded (/usr/lib/systemd/system/cockpit.socket; enabled; vendor preset: disabled)
   Active: active (listening) since Fri 2020-03-20 15:07:57 CST; 25min ago
     Docs: man:cockpit-ws(8)
   Listen: [::]:9090 (Stream)
    Tasks: 0
   Memory: 4.0K
   CGroup: /system.slice/cockpit.socket

3月 20 15:07:57 localhost.localdomain systemd[1]: Starting Cockpit Web Service Socket.
3月 20 15:07:57 localhost.localdomain systemd[1]: Listening on Cockpit Web Service Socket.
```

如果您正在系统上运行 firewalld，则需要使用如下命令打开防火墙中的 Cockpit 端口 9090。

```
#firewall-cmd --add-service=cockpit --permanent
#firewall-cmd --reload
```

```
[root@localhost ~]# firewall-cmd --add-service=cockpit --permanent
success
[root@localhost ~]# firewall-cmd --reload
success
```

3.4.3. Cockpit Web 控制台

Cockpit 使用 9090 端口，并且为 SSL 加密访问，通过浏览器登录，在网络浏览器中，通过以下 URL 打开 Cockpit 网络控制台：

本地：https://localhost:9090

远程：使用服务器的主机名或者 IP 地址，例如：

https://192.168.17.205:9090

登录时浏览器会提示链接不安全，点击添加例外。

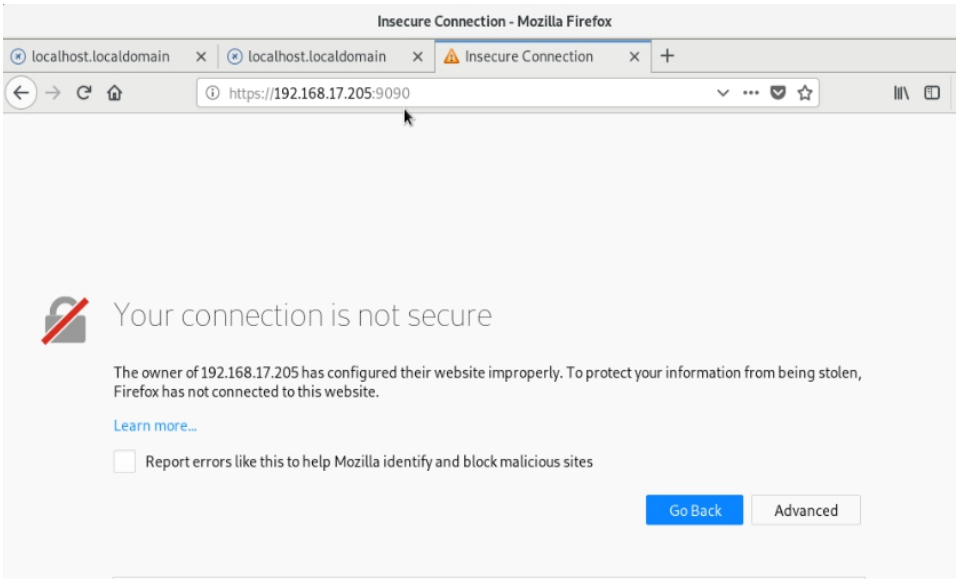


图 3-16 浏览器登录提示

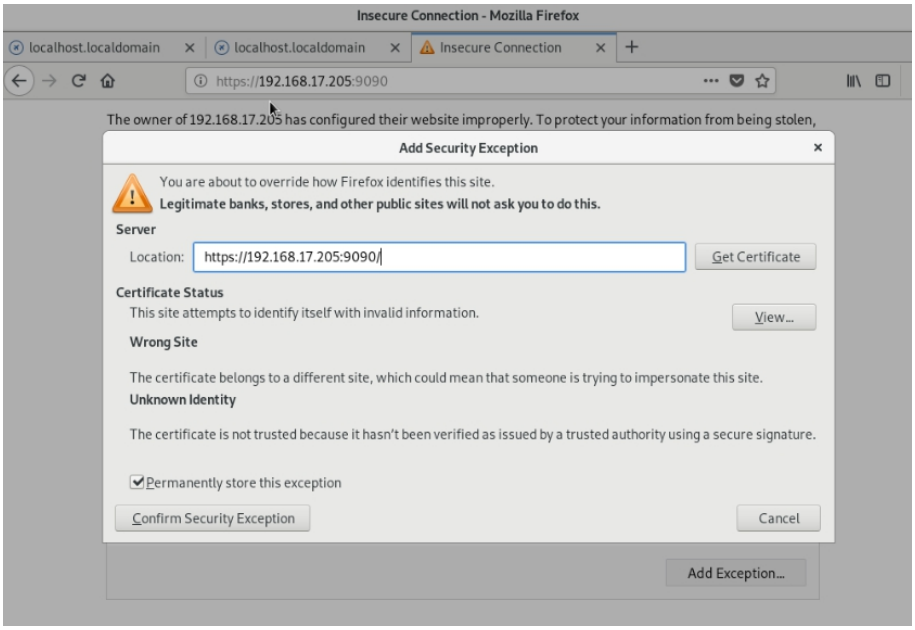


图 3-17 添加例外

3.4.3.1. 登录

在 Web 控制台登录屏幕中，输入系统用户名和密码进行登录，如图所示。如果用户帐户具有 sudo 特权，则可以执行管理任务，例如在 Web 控制台中安装软件，配置系统或配置 SELinux 等。

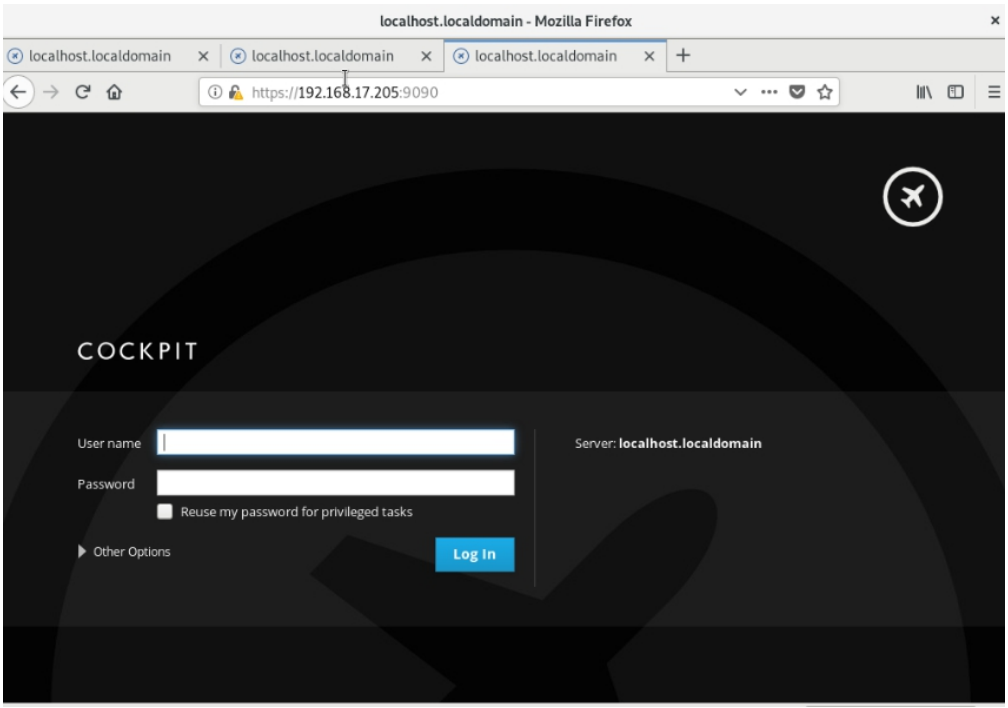


图 3-18 登录界面

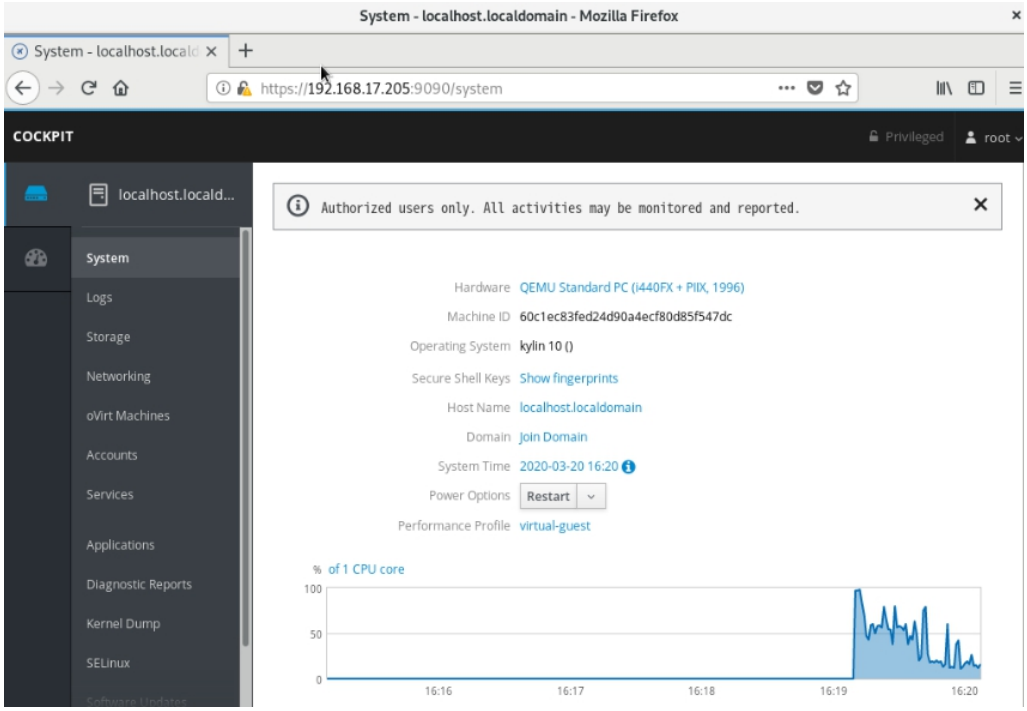


图 3-19 登录后界面

3.4.3.2. 语言选择

可以对语言进行切换，如图所示。

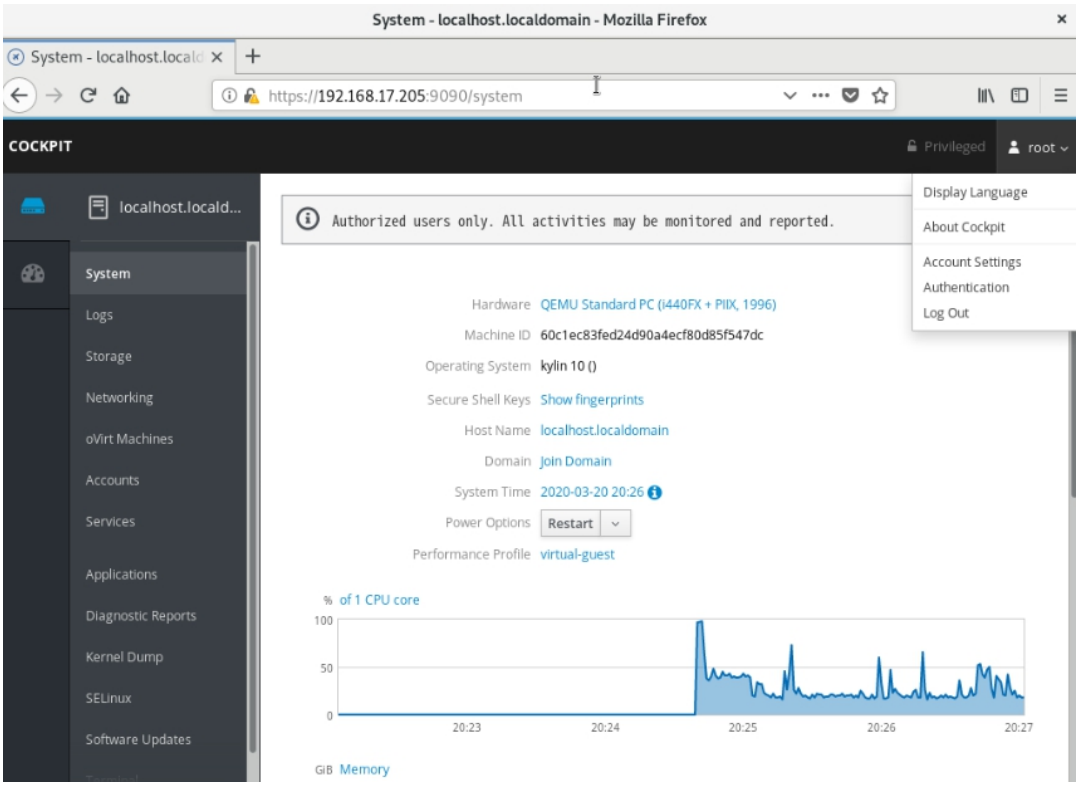


图 3-20 语言切换

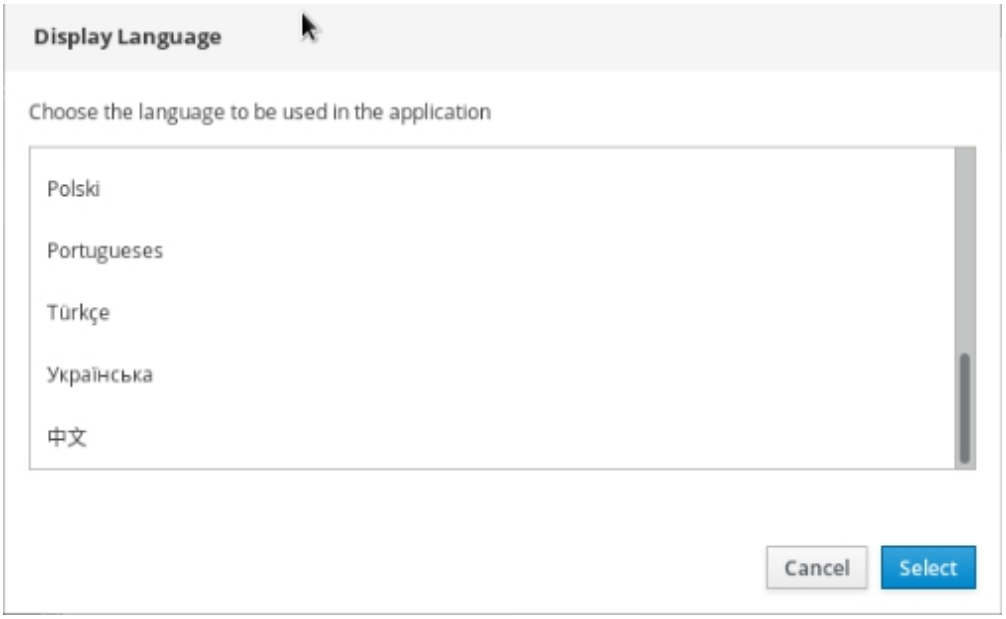


图 3-21 语言选择

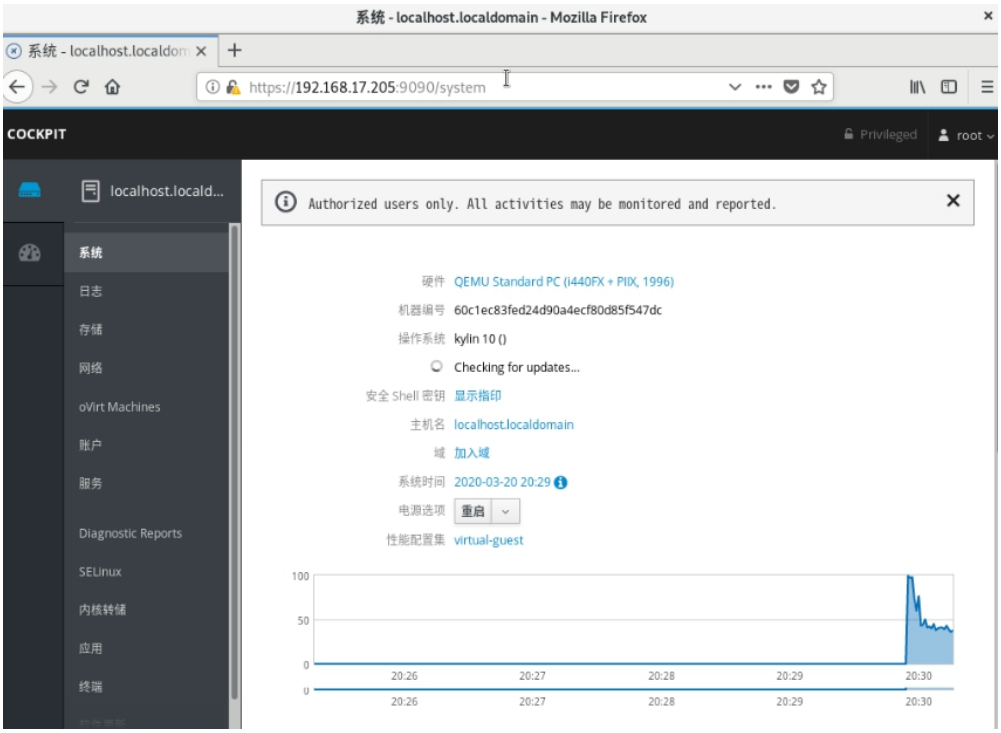


图 3-22 语言选定

3.4.3.3. 日志

Cockpit 将日志信息按时间和严重性来进行不同的分类。

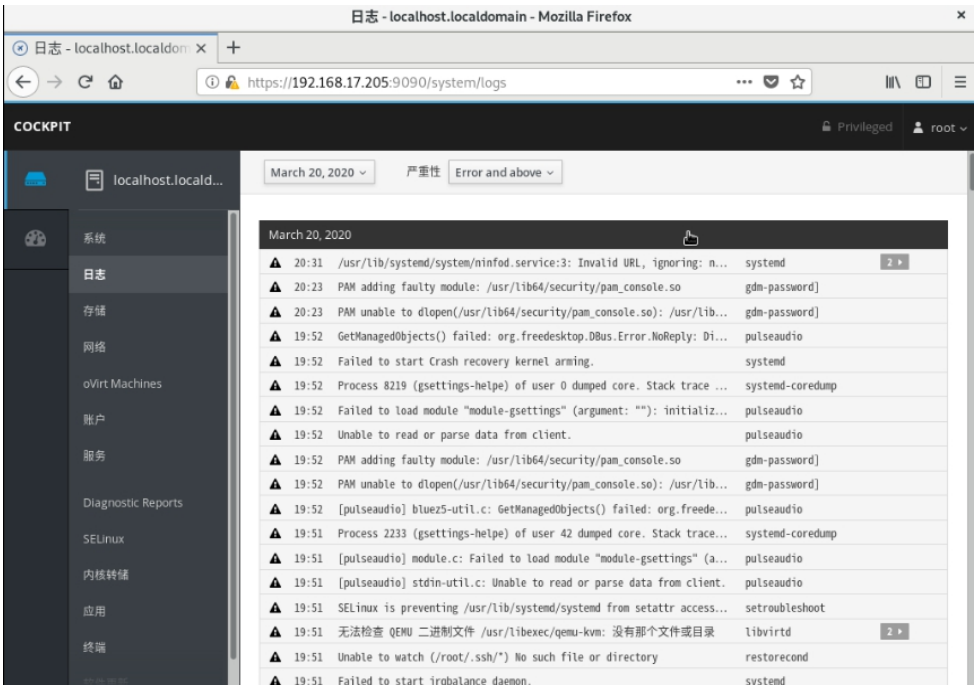


图 3-23 日志界面

3.4.3.4. 网络

网络页面可以看到两个可视化发送和接收速度的图。还可以看到可用网卡的列表，可以对网卡进行绑定设置，桥接，以及配置 VLAN。点击网卡就可以进行编辑操作。

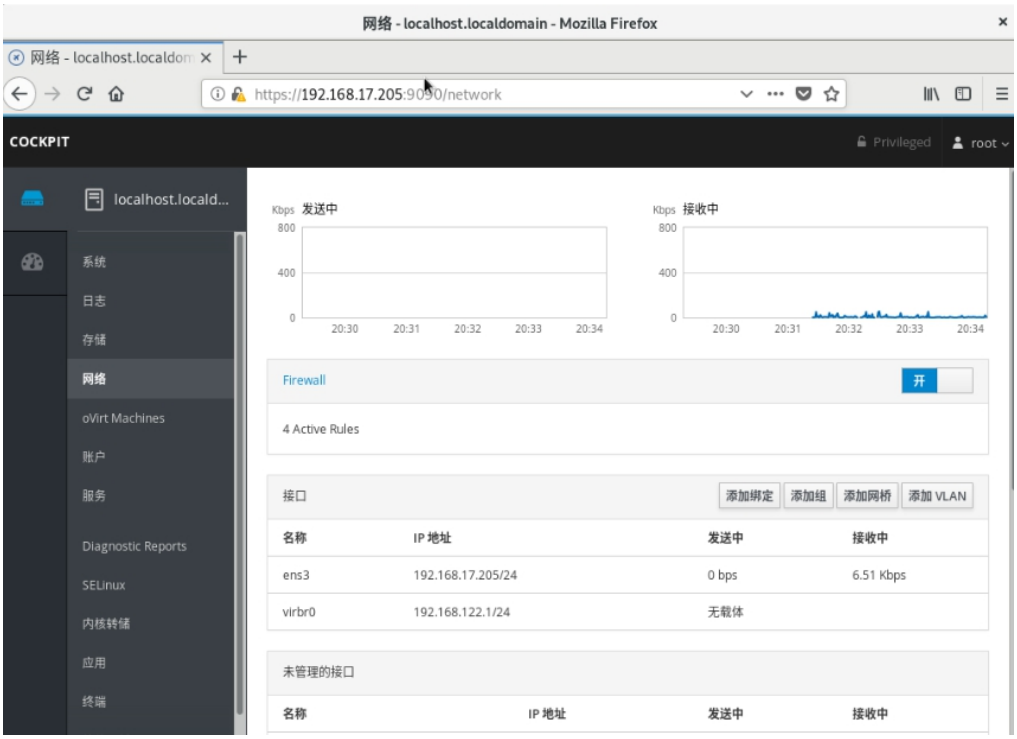


图 3-24 网络界面

3.4.3.5. 服务

服务被分成了 5 个类别：目标、系统服务、套接字、计时器和路径。

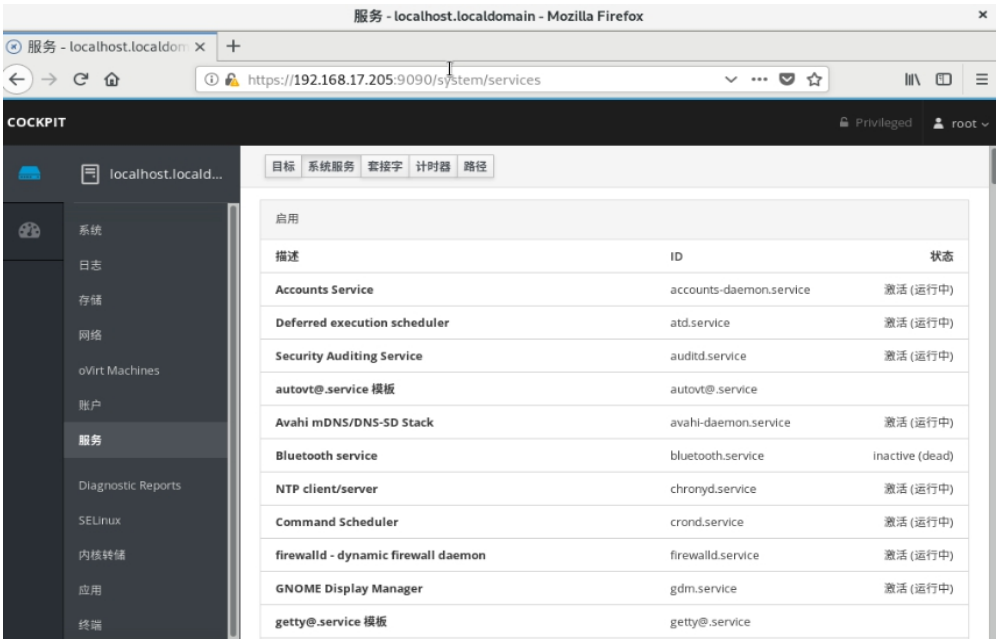


图 3-25 服务界面

3.4.3.6. 终端

Cockpit 界面提供实时终端执行任务，可以根据需求在 Web 界面和终端之间自由切换，可以快速执行任务，操作非常方便。

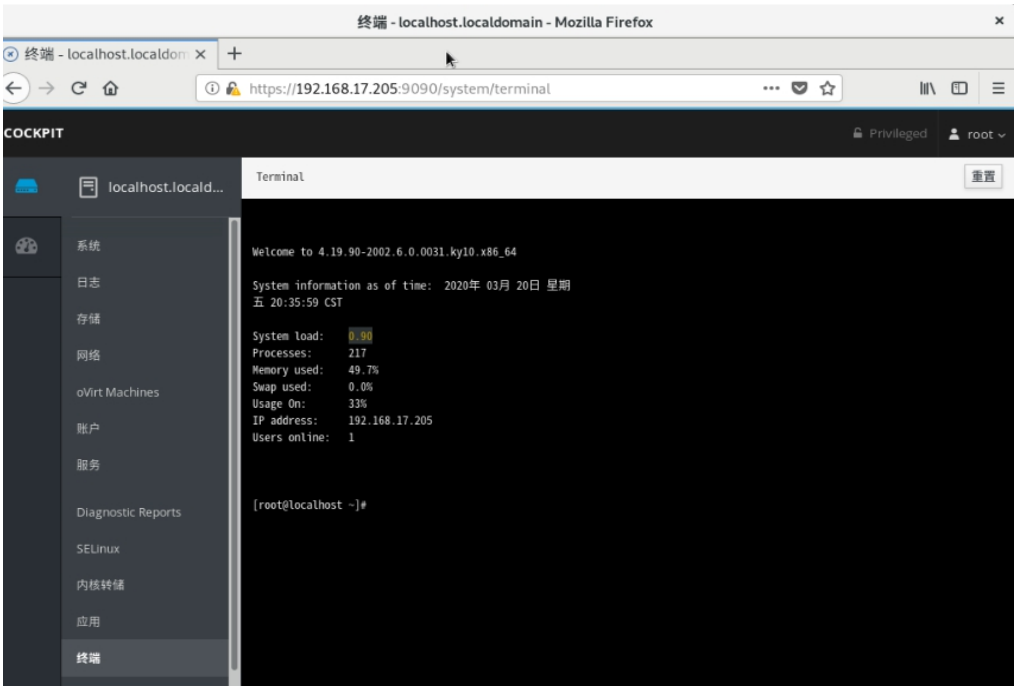


图 3-26 终端界面

3.4.3.7. 仪表盘

可以利用仪表盘进行多主机监控，点击右侧的“+”添加需要监控的主机。

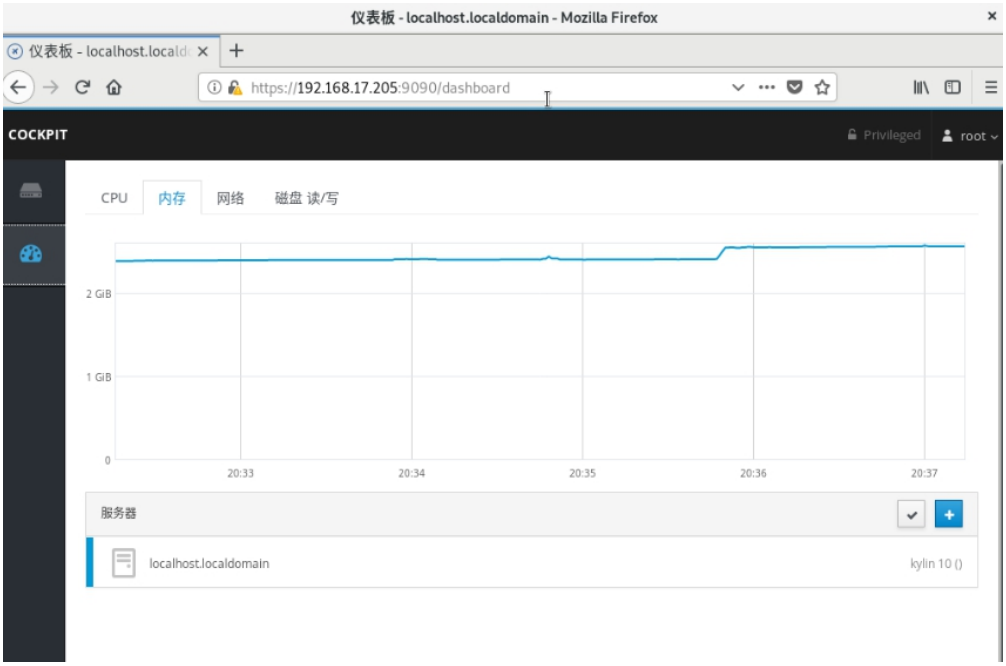


图 3-27 仪表盘界面

3.5. 系统日志

银河麒麟V10操作系统提供了一个基于图形用户界面的日志管理工具-麒麟日志查看器，使用该工具能够对于系统各方面的日志进行方便的查看、搜索、过滤和导出。

日志查看器具有以下特点:

智能化收集展示：实时同步收集展示系统内日志信息，根据日志类型进行归类显示。同时，具有过滤和聚合功能，对重复日志信息进行合并统计显示。

标准化全景态势：提供系统日志、启动日志、登录日志、应用日志以及安全日志。通过安全视角将事件标准化描述，包含目标事件等级、对象类型、时间、事件详细信息。

模块化维护扩展：采用模块化、可插拔架构设计，每类日志组件能够以模块化横向扩展，对不同类别日志独立维护，具有灵活易用、可维护特征。

详细内容请参考《银河麒麟日志查看器用户手册》。

第四章 安装和管理软件

dnf 是新一代的软件包管理器，首先出现在 Fedora 18 这个发行版中。而在 Fedora 22 中，它取代了 yum，正式成为了 Fedora 22 的包管理器。

dnf 包管理器克服了 yum 包管理器的一些瓶颈，提升了包括用户体验、内存占用、依赖分析和运行速度等多方面的内容。dnf 使用 rpm、libsolv 和 hawkey 库进行包管理操作，可以同 yum 同时使用。

4.1. 检查和升级软件包

4.1.1. 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：

```
[root@localhost ssh]# dnf check-update
上次元数据过期检查：2:09:08 前，执行于 2022年07月20日 星期三 09时54分49秒。

anaconda.x86_64                          33.19-31.p12.ky10                ks10-adv-os
anaconda-core.x86_64                     33.19-31.p12.ky10                ks10-adv-os
anaconda-tui.x86_64                      33.19-31.p12.ky10                ks10-adv-os
```

示例说明：

- PackageKit——软件包名称；
- x86_64——该软件包支持的 CPU 架构；
- 33.19-31.p12.ky10——可升级的软件包版本；
- ks10-adv-os——可升级的软件包所存储仓库。

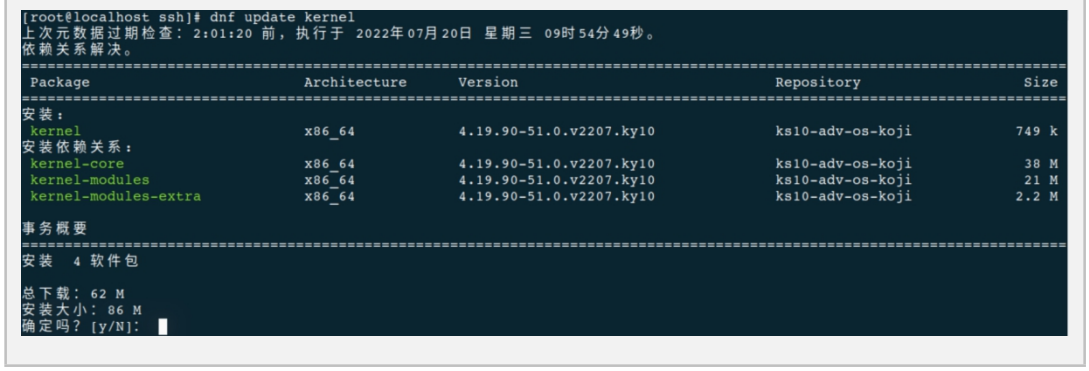
4.1.2. 升级软件包

dnf 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：

```
#dnf update {package_name}
```

升级 kernel 软件包命令为例：



上述输出的说明如下：

a) Package：用户需要下载升级的软件包和依赖软件包。

b) Architecture：该软件包所属的架构。

c) Version：软件包升级后的版本。

d) Repository：可升级软件包所属仓库。

e) Size：软件包大小。

f) dnf 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 dnf 命令时添加参数–y，效果等同于出现 Is this ok [y/N]: 时输入 yes。

g) 安装过程中如果出现错误导致安装过程终止，可以使用 dnf 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：

```
#dnf groupupdate 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：

```
#dnf update
```

4.1.3. 利用系统光盘与 dnf 离线升级系统

dnf 可与 yum 使用相同的配置文件，即配置 dnf 源可直接对/etc/yum.repos.d/中的.repo 文件进行编辑。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.repos.d/目录下：

```
#cp mount_dir/media.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 dnf 源并进行升级，以 root 用户执行：

```
#dnf 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 dnf 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
```

4.2. 管理软件包

dnf 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。

4.2.1. 检索软件包

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

```
#dnf search mesh
```

如果 dnf 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。

4.2.2. 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：

```
#dnf 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：

```
#dnf list glob_expression...
```

显示 abrt 相关软件包列表的命令如下：

```
#dnf 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：

```
#dnf list installed glob_expression...
```

显示包括 krb 的所有已安装软件包示例如下：

```
#dnf 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：

```
#dnf list available glob_expression...
```

显示所有可用的 gstreamer plug-ins 软件包列表：

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

查看软件仓库

成功注册后，可使用 dnf 来管理软件包。

查看可用的软件仓库可以使用以下命令：

```
#dnf repolist
```

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

```
#dnf repolist -v
```

```
#dnf 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：

```
#dnf repolist all
```

4.2.3. 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：

```
#dnf info package_name...
```

显示软件包 abrt 详细信息的命令：

```
#dnf info abrt
```

显示软件包 yum 详细信息的命令：

```
#dnf info yum
```

4.2.4. 安装软件包

用户可以以 root 用户使用以下命令安装软件包

```
#dnf install package_name
```

安装 sqlite 的 i686 架构的软件包示例：

```
#dnf 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：

```
#dnf install /usr/sbin/named
```

安装命令如下：

```
#dnf install httpd
```

如果要安装本地软件包，可以执行：

```
#dnf localinstall path
```

4.2.5. 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/N]:
. . .
```

输入 y，可以执行软件包下载。

4.2.6. 删除软件包

删除软件包可以执行以下命令：

```
dnf remove package_name...
```

删除 totem 软件包示例：

```
dnf remove totem
```

4.3. 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。

4.3.1. 软件包组列表

Summary 选项可以显示软件包可用组的数量：

```
dnf groups summary
```

以下为输出示例：

```
#dnf groups summary
可用组： 8
```

显示某个软件包组的全部信息可以用以下命令：

```
#dnf groups info glob_expression...
```

以下为 Server 组输出示例：

```
#dnf groups info Server
```

```
[root@localhost ~]# dnf groups info Server
Last metadata expiration check: 0:00:22 ago on Tue 09 Aug 2022 01:50:10 PM CST.
Environment Group: Server
 Description: An integrated, easy-to-manage server.
 Mandatory Groups:
   Base
   Container Management
   Core
   Hardware Support
   Headless Management
   Server product core
   Standard
 Optional Groups:
   Basic Web Server
   DNS Name Server
   Debugging Tools
   FTP Server
   File and Storage Server
   Hardware Monitoring Utilities
   Infiniband Support
   Mail Server
   Network File System Client
   Performance Tools
   Remote Management for Linux
   Virtualization Hypervisor
   Windows File Server
```

4.3.2. 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：

```
#dnf group list ids
```

查找开发软件包组列表的示例：

```
#dnf groups list ids deve\*
```

```
[root@localhost ~]# dnf group list ids deve\*
Last metadata expiration check: 0:12:05 ago on Tue 09 Aug 2022 01:50:10 PM CST.
Available Groups:
   Development Tools (development)
```

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。

```
#dnf group install "group name"
#dnf group install groupid
```

也可用通过以下两种命令安装：

```
#dnf install @group
#dnf install @^group
```

下面是 4 种安装开发工具软件分组的示例：

```
#dnf group install "Development Tools"
#dnf group install development
#dnf install @"Development Tools"
#dnf install @development
```

4.3.3. 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：

```
#dnf group remove group_name
#dnf 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：

```
#dnf remove @group
#dnf remove @^group
```

删除 KDE 桌面软件分组示例：

```
#dnf group remove "Development Tools"
#dnf group remove development
#dnf remove @"Development Tools"
#dnf remove @development
```

4.4. 软件包操作记录管理

dnf 可以使用 dnf history 命令进行管理操作。

4.4.1. 查看操作

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

```
#dnf history list 1..20
```

```
ID     | 命令行                                                  | 日期和时间       | 操作           | 更改
-------------------------------------------------------------------------------------------------
    20 | install libcap-devel                                    | 2021-06-08 14:02 | I, U           |    2
    19 | update yum                                              | 2021-05-31 14:01 | I, U           |    9 EE
    18 | install bind-utils                                      | 2021-05-28 09:57 | Install        |   10
    17 | install net-tools                                       | 2021-05-25 09:56 | Install        |    1
    16 | install docker                                          | 2021-05-24 15:15 | Install        |    2 EE
    15 | install zlib-devel                                      | 2021-05-24 13:29 | Install        |    1
    14 | install php-devel                                       | 2021-05-24 13:26 | Install        |    8
    13 | install telnet                                          | 2021-05-20 08:54 | Install        |    1
    12 | update                                                  | 2021-05-17 13:49 | E, I, U        |   17 EE
    11 | update                                                  | 2021-05-13 08:43 | Upgrade        |   13 EE
    10 | install php-mbstring                                    | 2021-04-30 17:33 | Install        |    2
     9 | install php-mysqli                                      | 2021-04-30 17:32 | Install        |    2
     8 | install php-json                                        | 2021-04-30 17:32 | Install        |    1
     7 | install php-xml                                         | 2021-04-30 17:32 | Install        |    1
     6 | install php-fpm                                         | 2021-04-30 17:25 | Install        |    2
     5 | install -y php                                          | 2021-04-30 17:23 | Install        |    3
     4 | install -y mariadb-server                               | 2021-04-30 16:01 | Install        |    6
     3 | install -y httpd                                        | 2021-04-30 15:55 | Install        |    7
     2 | update                                                  | 2021-04-30 15:46 | Upgrade        |   19 EE
     1 |                                                         | 2021-04-30 15:10 | Install        |  742 EE
```

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

```
#dnf history list start_id. . end_id
```

显示过去 5 条 dnf 信息示例如下：

```
#dnf history list 1..5
```

```
ID     | 命令行                                                  | 日期和时间       | 操作           | 更改
-------------------------------------------------------------------------------------------------
     5 | install -y php                                          | 2021-04-30 17:23 | Install        |    3
     4 | install -y mariadb-server                               | 2021-04-30 16:01 | Install        |    6
     3 | install -y httpd                                        | 2021-04-30 15:55 | Install        |    7
     2 | update                                                  | 2021-04-30 15:46 | Upgrade        |   19 EE
     1 |                                                         | 2021-04-30 15:10 | Install        |  742 EE
```

以上 dnf history list 输出显示内容说明如下：

ID——识别特定记录的标示数；

Command line——简要描述操作内容；

Date and time——该条记录的日期和时间；

Action(s)——描述操作类型；

Altered——记录操作影响的条目数。

下表是 Action 的不同说明：

表 4-1Action 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

4.4.2. 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：

```
#dnf history {id}
```

其中 id 是操作的 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：

```
#dnf 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：

```
#dnf history info start_id. . end_id
```

示例如下：

```
#dnf history info 4 . . 5
```

```
事务 ID:  4..5
起始时间     :  2021年04月30日 星期五 16时01分07秒
起始 RPM 数据库      :  747:8d43d8bfa0749f64ad69cd404fda7570feaaa531
结束时间       :  2021年04月30日 星期五 17时23分47秒 （82 分钟）
结束 RPM 数据库       : 756:44808f00897bf23cccd210933568326838a78c37
用户           :  root <root>
返回码     :  成功
Releasever     : 10
命令行     :  install -y mariadb-server
命令行     :  install -y php
注释         :
注释         :
已改变的包:
    安装 mariadb-common-3:10.3.9-8.ky10.x86_64     @ks10-adv-os
    安装 mariadb-errmessage-3:10.3.9-8.ky10.x86_64 @ks10-adv-os
    安装 mariadb-server-3:10.3.9-8.ky10.x86_64     @ks10-adv-os
    安装 mariadb-3:10.3.9-8.ky10.x86_64            @ks10-adv-os
    安装 perl-DBD-MySQL-4.046-6.ky10.x86_64        @ks10-adv-os
    安装 perl-DBI-1.642-2.ky10.x86_64              @ks10-adv-os
    安装 php-cli-7.2.10-12.ky10.x86_64             @ks10-adv-os
    安装 php-common-7.2.10-12.ky10.x86_64          @ks10-adv-os
    安装 php-7.2.10-12.ky10.x86_64                 @ks10-adv-os
```

4.4.3. 恢复与重复操作

如果想要撤销某个 dnf 操作，可以以 root 权限执行以下操作：

```
#dnf history undo {id}
```

如果需要重复某个 dnf 操作，可以以 root 权限执行以下操作：

```
#dnf history redo {id}
```

第五章 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 V10 系统。

5.1. 使用 systemd 管理系统服务

5.1.1. 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 5-1unit 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 5-2 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

5.1.1.1. 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 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。

5.1.1.2. 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。

5.1.2. 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 5-3service 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-all	systemctl list-units --type=service --all	显示所有服务状态

表 5-4 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

5.1.2.1. 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-unit-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
```

5.1.2.2. 显示服务状态

显示服务状态命令：

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

表 5-5 服务单元信息列表

文件	描述
Loaded	服务是否加载
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 firewalld.service 服务状态示例：

```
#systemctl status firewalld.service
```

```
[root@localhost ~]# systemctl status firewalld.service
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-08-04 09:01:39 CST; 4h 51min ago
     Docs: man:firewalld(1)
 Main PID: 1093 (firewalld)
    Tasks: 2
   Memory: 8.7M
   CGroup: /system.slice/firewalld.service
           └─1093 /usr/bin/python3 /usr/sbin/firewalld --nofork --nopid
```

显示 firewalld.service 服务启动前置依赖服务示例：

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

```
[root@localhost ~]# systemctl list-dependencies --after firewalld.service
firewalld.service
● ├─dbus.service
● ├─dbus.socket
● ├─polkit.service
● ├─system.slice
● ├─basic.target
● │ ├─-.mount
● │ ├─systemd-ask-password-plymouth.path
● │ ├─tmp.mount
● │ ├─paths.target
● │ │ ├─systemd-ask-password-console.path
● │ │ └─systemd-ask-password-wall.path
```

显示 firewalld.service 服务启动后导服务示例：

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

```
[root@localhost ~]# systemctl list-dependencies --before firewalld.service
firewalld.service
● ├─docker.service
● ├─multi-user.target
● │ ├─systemd-update-utmp-runlevel.service
● │ ├─graphical.target
● │ │ ├─systemd-update-utmp-runlevel.service
● │ │ └─shutdown.target
● │ └─shutdown.target
● ├─network-pre.target
```

5.1.2.3. 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
```

5.1.2.4. 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
```

5.1.2.5. 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
```

5.1.2.6. 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.service
Created symlink
/etc/systemd/system/multi-user.target.wants/httpd.service →
 /usr/lib/systemd/system/httpd.service.
```

5.1.2.7. 禁用服务

禁用服务命令：

```
#systemctl disable name.service
```

禁用 bluetooth 服务示例：

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

service.

5.1.3. 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 5-6 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 5-7 SysV 初始化命令与 systemctl 对比

旧命令	新命令	描述
runlevel	systemctl list-units --type target	显示当前目标
telinit runlevel	systemctl isolate name.target	变更当前目标

5.1.3.1. 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
```

5.1.3.2. 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
```

5.1.3.3. 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
Remove /etc/systemd/system/default.target
Created symlink
/etc/systemd/system/default.target → /usr/lib/systemd/syste
m/multi-user.target.
```

5.1.3.4. 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
```

5.1.3.5. 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：

```
#systemctlrescue
```

5.1.3.6. 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
```

5.1.4. 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：

```
#systemctl -H root@server-01.example.com status
httpd.service
```

5.1.5. 创建和修改 systemd 单元文件

5.1.5.1. 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 5-8 【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 URL 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，之前的配置不再起作用。
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=是相互独立的，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，如果配置了某些项，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（两者都是必需的）或修改以修复它（两者之一或两者都不是必需的），后一种情况，会将不需要的依赖删除，或停止冲突。

表 5-9【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systemctlstop|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。
RemainAfterExit	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后，是否依然将此服务视为活动(active)状态。

表 5-10 【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，会建立相应的链接。
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。EnvironmentFile 表述预定义的该服务的环境变量，**PIDFile** 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。

5.1.5.2. 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，确保所有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-documentingtextedi
tor
[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 文件为 sshd-second.config

```
#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/usr/lib/systemd/system/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
```

在执行脚本里，添加第二个 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。

5.1.5.3. 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
```

5.2. 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。

银河麒麟高级服务器操作系统包含了通用的 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。

5.2.1. SSH 协议

5.2.1.1. 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。

5.2.1.2. 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。

5.2.1.3. 协议版本

SSH 目前存在两个版本：版本 1 和较新的版本 2。银河麒麟高级服务器操作系统中的 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。

5.2.2. 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。

5.2.2.1. 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。

5.2.2.2. 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。

5.2.2.3. 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。

5.2.3. 配置 OpenSSH

5.2.3.1. 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 5-11 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 5-12 特定用户的配置文件。

表 5-11 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

文件	描述
	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。

文件	描述
/etc/sysconfig/sshd	sshd 服务的配置文件。

表 5-12 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。
~/.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。

5.2.3.2. 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
Created symlink
/etc/systemd/system/multi-user.target.wants/sshd.service →
/usr/lib/systemd/system/sshd.service.
```

5.2.3.3. 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。

5.2.3.4. 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基于密钥的登录过程是否可用后再配置 PubkeyAuthentication 为 no。

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 5.2.3.5 生成秘钥对章节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。

银河麒麟高级服务器操作系统 V10 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。

5.2.3.5. 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/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。

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。

5.2.3.6. 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 4.2.4 安装软件包。了解如何在银河麒麟高级服务器操作系统中安装新的包）。

5.2.3.7. 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 SHA256:Ixy64icRYc/h7XSOvUVywS7t7ThtmOsPT1s07wDD5P8.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

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
SHA256:b0gGbl+Xk/l+Ve76j3mqpYSty0n3gR9QiIsd+8oV3Gl
no comment (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，验证服务端没有受到攻击。

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 release V10 (Lance)
```

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。

5.2.3.8. 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 的远程主机用户家目录上，可以在 shell 命令行提示符下输入以下命令：

```
#scp taglist.vim USER@penguin.example.com:~
```

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

```
$scp -r .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
```

5.2.3.9. 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 5– 13）。

表 5-13 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。

命令	描述
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。

5.2.4. 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。

5.2.4.1. 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：

```
#dnf group install "X Window System"
```

要了解关于软件包分组的更多信息，请参见 4.2 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。

5.2.4.2. 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。

5.3. 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：

相比以往银河麒麟高级服务器操作系统,银河麒麟高级服务器操作系统 V10 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。

5.3.1. 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。

5.3.1.1. 安装 VNC 服务端

要安装 TigerVNC 服务端，请以 root 用户执行以下命令：

```
#dnf install tigervnc-server
```

5.3.1.2. 配置 VNC 服务端

VNC 服务端可以被配置了为一个或多个用户（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。

(1) 为指定用户复制 vncserver@.service 文件到/etc/systemd/system 目录

(以 smbuser 用户为例)

```
#cp /usr/lib/systemd/system/vncserver@.service
/etc/systemd/system/vncserver-smbuser@.service
```

(2) 修改 service 文件将<USER>替换成实际的用户

```
#vim /etc/systemd/system/vncserver-smbuser@.service
```

```
#The vncserver service unit file
#
#Quick HowTo:
#1. Copy this file to /etc/systemd/system/vncserver@.service
#2. Replace <USER> with the actual user name and edit vncserver
#    parameters appropriately
#3. Run `systemctl daemon-reload`
#4. Run `systemctl enable vncserver@:<display>.service`
#
#DO NOT RUN THIS SERVICE if your local area network is
#untrusted!   For a secure way of using VNC, you should
#limit connections to the local host and then tunnel from
#the machine you want to view VNC on (host A) to the machine
#whose VNC output you want to view (host B)
#
#[user@hostA ~]#ssh -v -C -L 590N:localhost:590M hostB
#
```

#this will open a connection on port 590N of your hostA to hostB's port 590M
#(in fact, it ssh-connects to hostB and then connects to localhost (on hostB).
#See the ssh man page for details on port forwarding)
#
#You can then point a VNC client on hostA at vncdisplay N of localhost and with
#the help of ssh, you end up seeing what hostB makes available on port 590M
#
#Use "-nolisten tcp" to prevent X connections to your VNC server via TCP.
#
#Use "-localhost" to prevent remote VNC clients connecting except when
#doing so through a secure tunnel. See the "-via" option in the
#`man vncviewer' manual page.

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

[Service]
Type=forking
WorkingDirectory=/home/smbuser
User=smbuser
Group=smbuser

PIDFile=/home/smbuser/.vnc/%H%i.pid

ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'

```
ExecStart=/usr/bin/vncserver -autokill %i
ExecStop=/usr/bin/vncserver -kill %i

Restart=on-success
RestartSec=15

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

(3) 执行 systemctl daemon-reload

5.3.1.3. 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 5.3.1.2 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-USER_1@: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-USER_1@: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
```

5.3.1.4. 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-USER_1@: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-USER_1@:display_number.service
```

5.3.2. 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：

```
#dnf 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 5.3.3.2 为 VNC 配置防火墙。

5.3.3. 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。

5.3.3.1. 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：

```
#dnf 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。

5.3.3.2. 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。

5.3.3.3. 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
```

第六章 服务器

6.1. 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。

6.1.1. Apache HTTP 服务器

银河麒麟高级服务器操作系统中提供的可用 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。

6.1.1.1. 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

在银河麒麟高级服务器操作系统的早期版本中，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 。

在银河麒麟高级服务器操作系统中，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://localhost/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/apxs。

被删除模块

在银河麒麟高级服务器操作系统中，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。

6.1.1.2. 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 6.1.1.8 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 http://httpd.apache.org/docs/2.4/upgrading.html 。

6.1.1.3. 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。

```
#dnf install httpd
```

通常，在银河麒麟高级服务器操作系统中，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 5.1 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动：

```
#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。

在银河麒麟高级服务器操作系统中，有关如何管理系统服务的更多信息，请参看 5.1 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
```

6.1.1.4. 编辑配置文件

默认情况下，当启动 httpd 时，会读取表 6-1 httpd 服务配置文件中列出的 httpd 服务配置文件。

表 6-1 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 6.1.1.3 运行 http 的服务中的“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。

6.1.1.5. 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。在银河麒麟高级服务器操作系统中，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 6.1.1.3 运行 http 的服务中的“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。执行以下命令完成安装。

```
#dnf 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。

6.1.1.6. 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/httpd-vhosts.conf 配置文件到/etc/httpd/conf.d/目录下，并替换“@@Port@@”和“@@ServerRoot@@”为相应服务端口和 web 根目录。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/penguin.example.com-error_log"
    CustomLog  "/var/log/httpd/penguin.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 6.1.1.3 运行 http 的服务中的“重启服务”。

6.1.1.7. 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。银河麒麟高级服务器操作系统也支持基于 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 6-2 查看浏览器中通常可用的 CA。

表 6-2 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。

6.1.1.8. 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：

```
#dnf 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 6.1.1.3 运行 http 的服务中的“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。

6.1.1.9. 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 6.1.1.9 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 6.1.1.3 运行 http 的服务中的重启服务。

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
```

6.1.1.10. 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 OpenSSL 软件包。由 root 用户执行以下命令可以完成安装：

```
#dnf install openssl
```

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 用户用以下命令为 hostname.key 重新生成证书：

```
#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，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
```

6.1.1.11. 为 HTTP 和 HTTPS 配置防火墙

默认情况下，银河麒麟高级服务器操作系统是不启用 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
```

6.2. 目录服务器

6.2.1. 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。

6.2.1.1. 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。

6.2.1.2. LDAP 术语

下面列出在本章中使用 LDAP-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。

6.2.1.3. 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。

6.2.1.4. 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 6.2.2 安装 OpenLDAP 组件；
2) 配置参见 6.2.3 配置 OpenLDAP 服务器；
3) 启动 slapd 服务，参见 6.2.5 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。

6.2.2. 安装 OpenLDAP 组件

OpenLDAP 的函数库和工具由以下包提供：

表 6-3 OpenLDAP 包列表

包	描述
openldap	该包包含运行 OpenLDAP 服务器和客户端应用所必须的函数库。
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。
openldap-devel	开发包。
migrationtools	通过 migrationtools 实现 OpenLDAP 用户及用户组的添加，导入系统账户。

此外，以下包通常和 LDAP 服务器使用：

表 6-4 常用附加 LDAP 包列表

包	描述
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 dnf 安装以下包：

```
dnf install package...
```

例如，安装 LDAP 服务器

```
#dnf 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“4.2.4 安装软件包”。

6.2.2.1. 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 6-5 OpenLDAP 服务端工具列表

命令	描述
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。创建 OpenLDAP 目录索引，提高查询效率。
slappasswd	允许您创建一个加密的用户密码，为 ldapmodify 组件，或者用在 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。

6.2.2.2. 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 6-6 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。
ldapdelete	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。

6.2.2.3. 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,但是银河麒麟高级服务器操作系统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，Thunderbird ，Evolution，或者 Ekiga。

6.2.3. 配置 OpenLDAP 服务器

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

表 6-7 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 不再使用/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。

6.2.3.1. 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 6-8 可用的 olcAllows 选项。默认的选项是 bind_v2。

表 6-8 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 6-9 可用的 olcDisallows 选项。没有默认不可使用的特性。

表 6-9 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
```

6.2.3.2. 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
```

6.2.3.3. 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。

6.2.3.4. 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，在银河麒麟高级服务器操作系统中 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 olcTLSCACertificatePath 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。

6.2.3.5. 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
```

olcSyncrepl

olcSyncrepl 使用 sync 备份模式

```
olcSyncrepl on
```

6.2.3.6. 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。

6.2.4. 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 选项是这次设置在系统重启一直保持有效。

6.2.5. 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。

6.2.5.1. 启动服务

使用 root 权限，开启 slapd 服务，命令如下：

```
#systemctl start slapd.service
```

使用 root 权限，设置开机启动 slapd 服务，命令如下：

```
#systemctl enable slapd.service
Created symlink /etc/systemd/system/openldap.service → /usr/lib/systemd/system/slapd.service.
Created symlink /etc/systemd/system/multi-user.target.wants/slapd.service → /usr/lib/systemd/system/slapd.service.
```

6.2.5.2. 停止服务

停止服务，命令如下：

```
#systemctl stop slapd.service
```

设置开机不启动服务，命令如下：

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

6.2.5.3. 重启服务

重启服务，命令如下：

```
#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。

6.2.5.4. 检查运行状态

检查 slapd 服务运行状态，命令如下：

```
#systemctl is-active slapd.service
active
```

6.2.6. 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 6.2.2 安装 OpenLDAP 组件和 6.2.3 配置 OpenLDAP 服务器。在客户端上，输入命令如下：

```
#dnf install openldap openldap-clients nss-pam-ldapd
```

6.2.6.1. 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：

```
#dnf 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

```
# 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 6-10 常用的 LDAP 迁移脚本。

表 6-10 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。

6.3. 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，银河麒麟高级服务器操作系统的 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。

6.3.1. 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

```
#dnf install samba
```

6.3.1.1. 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。

6.3.1.2. 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。

6.3.1.3. 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
```

6.3.1.4. 启动和关闭 Samba

启动命令如下：

```
#systemctl start smb.service
```

关闭命令如下：

```
#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，命令如下：

```
#systemctl enable smb.service
```

6.3.1.5. 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
...
```

3) 修改/etc/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。

6.3.1.6. 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 6.3.1.3 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
```

6.3.1.7. Samba Distribution Programs

net 命令：

```
    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 命令：

```
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 命令：

```
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 命令：

```
    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 命令：

```
    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 命令：

```
    smbclient <//server/share> <password> <options>
```

smbclient 是 UNIX 系统通用的客户端，功能和 ftp 类似。

smbcontrol 命令：

```
    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 命令：

```
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 命令：

```
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 命令：

```
smbstatus <options>
```

smbstatus 项目显示当前 Samba 连接状态

smbtar 命令：

```
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 命令：

```
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 命令：

```
    wbinfo <options>
```

wbinfo 程序显示 winbindd 守护进程信息。winbindd 进程必须是运行的。

6.3.1.8. 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

6.3.2. 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 5.2OpenSSH。

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,它可以使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。

6.3.2.1. vsftpd 服务器

vsftpd 设计为快、稳定和安全。vsftpd 是银河麒麟高级服务器操作系统中的 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/va/ftp/目录是共享目录，vsftpd 指定/va/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

启动命令：

```
#systemctl start vsftpd.service
```

停止命令：

```
#systemctl stop vsftpd.service
```

重启命令：

```
#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
Created symlink
/etc/systemd/system/multi-user.target.wants/vsftpd.service →
/usr/lib/systemd/system/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，命令如下：

```
#systemctl start vsftpd.target
#systemctl enable vsftpd.target
Created symlink
/etc/systemd/system/multi-user.target.wants/vsftpd.target →
/usr/lib/systemd/system/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

修改 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，在银河麒麟高级服务器操作系统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，在银河麒麟高级服务器操作系统中这个是缺省的设置，ftpd_is_daemon 必须设置为 1。

6.3.3. 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银河麒麟高级服务器操作系统版本，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**开始->控制面板->打印机，**打印机配置工具启动如下图。

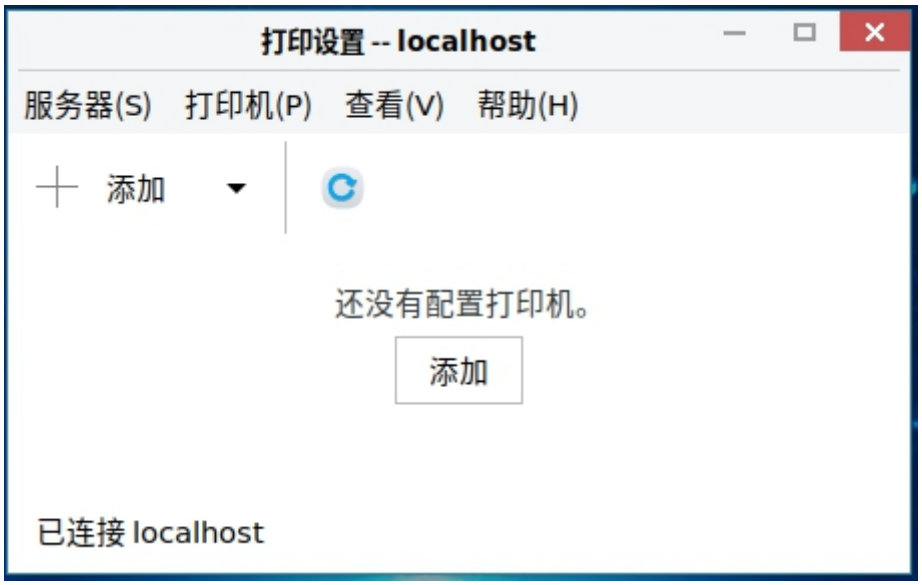


图 6-1 打印设置

6.4. 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。

6.4.1. 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。

6.4.1.1. 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;
- chronyd 提供支持工作的收益或损失的速度实时时钟,硬件时钟,维护计算机是关闭的时候。它可以使用这些数据在系统启动时设置系统时间使用调整后的实时时钟的时间的价值。写作时,仅可在 Linux。

  ntpd 可以做 chronyd 不能做的事情。
- ntpd 支持 NTP 版本 4（RFC5905），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(RFC5906)。chronyd 使用 NTP 版本 3（RFC1305），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。

6.4.1.2. 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。

6.4.2. 理解 CHRONY 及其配置

6.4.2.1. 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。

6.4.2.2. 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。

注释

注释前应以#、%或！开头。

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 cmdallow 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。

6.4.3. 使用 chrony

6.4.3.1. 安装 chrony

使用 root 用户登录

```
#dnf install chrony
```

默认的 chrony 进程位置/usr/sbin/chronyd。命令行组件安装在/usr/bin/chronyc。

6.4.3.2. 检查 chronyd 状态

```
#systemctl status chronyd
```

6.4.3.3. 启动 chronyd

启动命令：

```
#systemctl start chronyd
```

设置开机自启动命令：

```
#systemctl enable chronyd
```

6.4.3.4. 关闭 chronyd

关闭命令：

```
#systemctl stop chronyd
```

取消开机自启动命令：

```
#systemctl disable chronyd
```

6.4.3.5. 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

与之进行同步的 ntp 服务器的参考 ID（一串 16 进制数字）和名称（或 ip 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

偏移量的长期平均值。

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

频率上的估计误差。

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。

Root dispersion

通过所有经过的计算机，回到最终同步的 stratum-1 计算机累积的总弥散。

下面的公式给出了计算机时钟精度的绝对界（假设 stratum-1 计算机的时间是正确的）：clock_error <= |system_time_offset| + root_dispersion + (0.5 * root_delay)。

Update interval

最后两次时钟更新的时间间隔。

Leap status

可能值为 Normal、Insert second、Delete second、Not synchronized。

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。

6.4.3.6. 手动调整系统时钟

命令如下：

```
#chronyc
chrony> password commandkey-password
200 OK
chrony> makestep
200 OK
```

6.4.4. 为不同的环境设置 chrony

6.4.4.1. 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
```

6.4.4.2. 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。

6.4.5. 使用 chronyc

6.4.5.1. 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
```

6.4.5.2. 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。

第一条命令必须输入密码：

```
chronyc> password password
200 OK
```

密码不允许有空格。

假如密码不是 MD5 哈希，哈希密码必须被 authhash 命令在前。

```
chronyc> authhash SHA1
chronyc> password
HEX:A6CFC50C9C93AB6E5A19754C246242FC5471BCDF
200 OK
```

6.5. 配置 NTP 使用 NTPD

6.5.1. 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。

6.5.2. 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 JJY-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。

6.5.3. 理解 NTP

银河麒麟高级服务器操作系统使用的 NTP 版本，在 RFC 1305 Network 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。

6.5.4. 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。

6.5.5. 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。

6.5.6. NTP 身份验证选项

在当前网络，攻击者可以通过发送 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。

6.5.7. 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。在银河麒麟高级服务器操作系统中，kvm 默认使用 kvm-clock。

6.5.8. 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。

6.5.9. 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。

6.5.10. 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。

6.5.11. 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
```

6.5.12. 检查 NTP 守护进程是否安装

命令如下：

```
#dnf install ntp
```

6.5.13. ntpd 的安装

```
#dnf install ntp
```

设置 ntpd 开机启动：

```
#systemctl enable ntpd
```

6.5.14. 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
```

6.5.15. 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。

6.5.15.1. 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。

6.5.15.2. 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。

6.5.16. 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
```

在银河麒麟高级服务器操作系统系列，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。

6.5.17. 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。

6.5.17.1. 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。

6.5.17.2. 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
```

6.5.17.3. 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。

6.5.17.4. 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。

6.5.17.5. 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
```

6.5.17.6. 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。

6.5.17.7. 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。

6.5.17.8. 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。

6.5.17.9. 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。

6.5.17.10. 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
```

6.5.17.11. 配置 iburst 选项

在服务器命令结尾添加该选项：

```
iburst
```

6.5.17.12. 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
```

6.5.17.13. 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
```

6.5.17.14. 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
```

6.5.17.15. 为 NTP 数据包配置 Time-to-Live

在 server 或 peer 命令后添加该选项：

```
ttl value
```

6.5.17.16. 配置 NTP 使用版本

在 server 或 peer 命令后添加该选项：

```
version value
```

6.5.18. 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
```

6.5.19. 配置时钟源

在系统列出可用的时钟源：

```
#cd /sys/devices/system/clocksource/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
```

6.6. 使用 ptp4l 配置 PTP

6.6.1. 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。银河麒麟高级服务器操作系统 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)。

6.6.1.1. 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。

6.6.2. 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。

6.6.2.1. 检查驱动和硬件支持

使用 ethtool 命令查询：

```
#ethtool -T eth3
```

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

6.6.2.2. 安装 PTP

```
#dnf install linuxptp
```

这将会安装 ptp4l 和 phc2sys。

6.6.2.3. 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[81921.804]: selected /dev/ptp0 as PTP clock
ptp4l[81921.805]: driver changed our HWTSTAMP options
ptp4l[81921.806]: tx_type    1 not 1
ptp4l[81921.806]: rx_filter 1 not 12
ptp4l[81921.806]: port 1: INITIALIZING to LISTENING on
INIT_COMPLETE
ptp4l[81921.806]: port 0: INITIALIZING to LISTENING on
INIT_COMPLETE
ptp4l[81929.604]: port 1: LISTENING to MASTER on
ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
```

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。

6.6.3. 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
```

6.6.4. 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
```

6.6.5. 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
```

6.6.6. 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。

6.6.7. 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
INITIALIZE
ptp4l[352.361]: port 0: INITIALIZING to LISTENING on
INITIALIZE
.....
```

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    2729
    phc2sys[529.528]: phc offset       54658 s1 freq   -37690
delay    2725
    .......
```

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
INITIALIZE
    .......
```

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
```

6.6.8. 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 phc2sys 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
```

6.6.9. 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 -l 0.0001
```

6.6.10. 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 chronyd 或 ntpd 配置文件，启动 ptp4l，phc2sys 和 phc2sys 或 ntpd。

6.6.10.1. 作为一个服务启动 timemaster

使用 root 用户执行如下命令：

```
#systemctl start timemaster
```

启动时，会读取/etc/timemaster.conf 配置文件。

6.6.10.2. 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。

6.6.10.3. 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.confentry 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.confentry 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
```

6.6.11. 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
```

第七章 监控和自动化

7.1. 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银河麒麟服务器操作系统中检索这类信息。

7.1.1. 查看系统进程

7.1.1.1. 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：

```
[root@localhost /]# ps ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:00 /usr/lib/systemd/systemd --switched-root
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
      4 ?        I<     0:00 [rcu_par_gp]
      6 ?        I<     0:00 [kworker/0:0H-kblockd]
      8 ?        I<     0:00 [mm_percpu_wq]
     10 ?        S      0:00 [ksoftirqd/0]
     11 ?        I      0:00 [rcu_sched]
     12 ?        I      0:00 [rcu_bh]
     13 ?        S      0:00 [migration/0]
     14 ?        S      0:00 [cpuhp/0]
     16 ?        S      0:00 [kdevtmpfs]
     17 ?        I<     0:00 [netns]
```

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。

7.1.1.2. 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 7-1 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。

7.1.1.3. 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。**开始->所有程序->系统工具->系统监视器**点击打开，如下图：

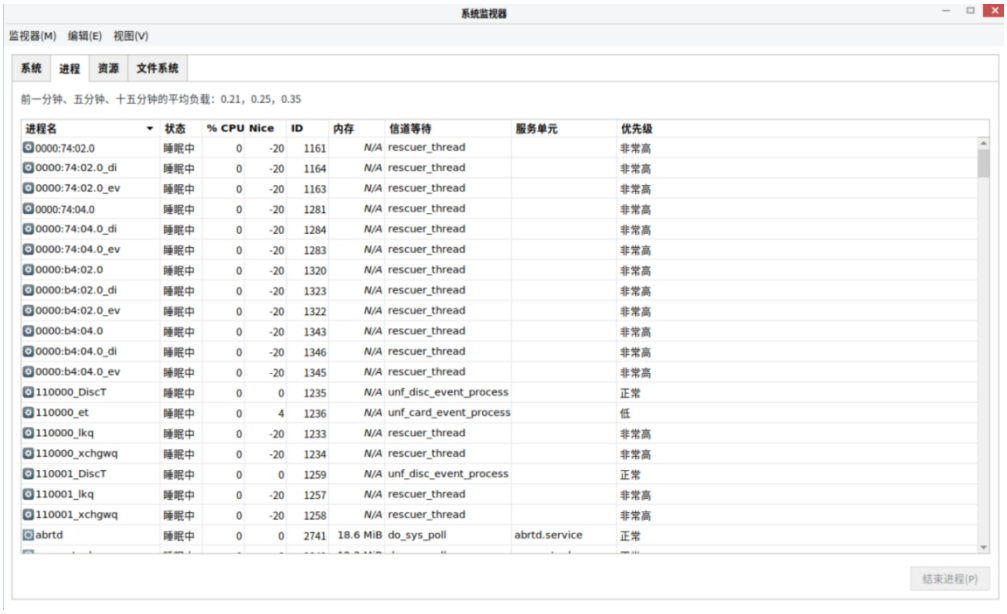


图 7-1 进程

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。

7.1.2. 查看内存使用情况

7.1.2.1. 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
```

7.1.2.2. 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，点击资源标签页查看系统内存使用情况。

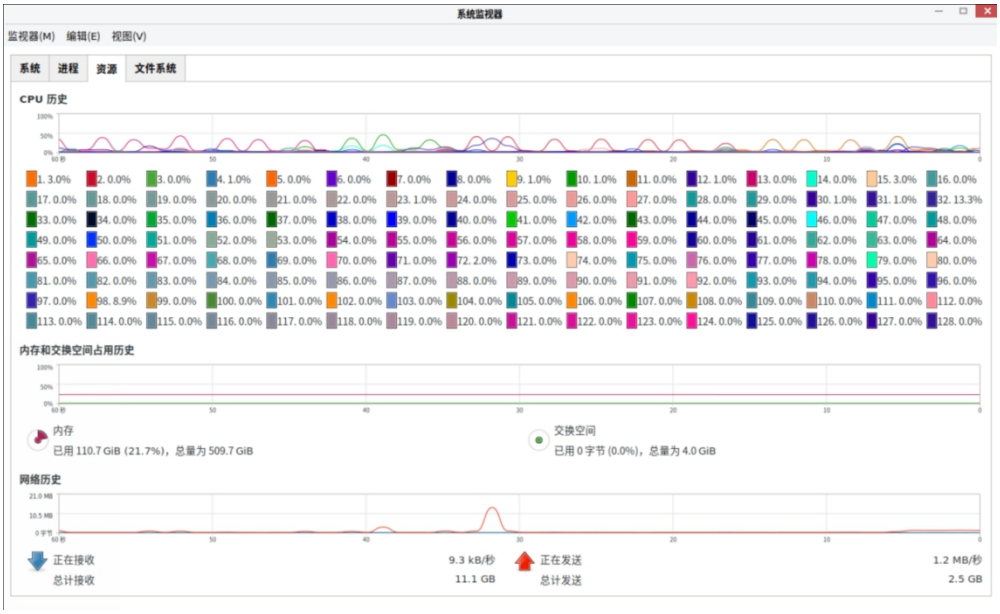


图 7-2 资源

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。

7.1.3. 查看 CPU 使用

7.1.3.1. 使用系统监控工具

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 当前使用率的图表查看块设备和文件系统。

7.1.3.2. 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。

7.1.3.3. 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-aebcad2ae824"
    TYPE="ext4"
    /dev/mapper/vg_kvm-lv_swap:
UUID="d7ef54ca-9c41-4de4-ac1b-4193b0c1ddb6"
    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 root 权限。

```
    blkid -po udev /dev/vda1
```

例如：

```
    #blkid -po udev /dev/vda1
    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。

7.1.3.4. 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
TARGET                  SOURCE     FSTYPE  OPTIONS
/sys                    sysfs      sysfs   rw,nosuid,nodev,noexec,relatime
/proc                   proc       proc    rw,nosuid,nodev,noexec,relatime
/dev                    devtmpfs   devtmpf rw,nosuid,size=997904k,nr_inodes=2
/sys/kernel/security    securityfs securit rw,nosuid,nodev,noexec,relatime
/dev/shm                tmpfs      tmpfs   rw,nosuid,nodev
/dev/pts                devpts     devpts  rw,nosuid,noexec,relatime,gid=5,mo
```

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:

```
[root@localhost /]# findmnt -t xfs
TARGET  SOURCE               FSTYPE OPTIONS
/       /dev/mapper/klas-root xfs    rw,relatime,attr2,inode64,noquota
└─/boot /dev/sda1             xfs    rw,relatime,attr2,inode64,noquota
```

有关可用命令行选项的完整列表，查看 findmnt(8) man 手册。

7.1.3.5. 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。

7.1.3.6. 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。

7.1.3.7. 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。

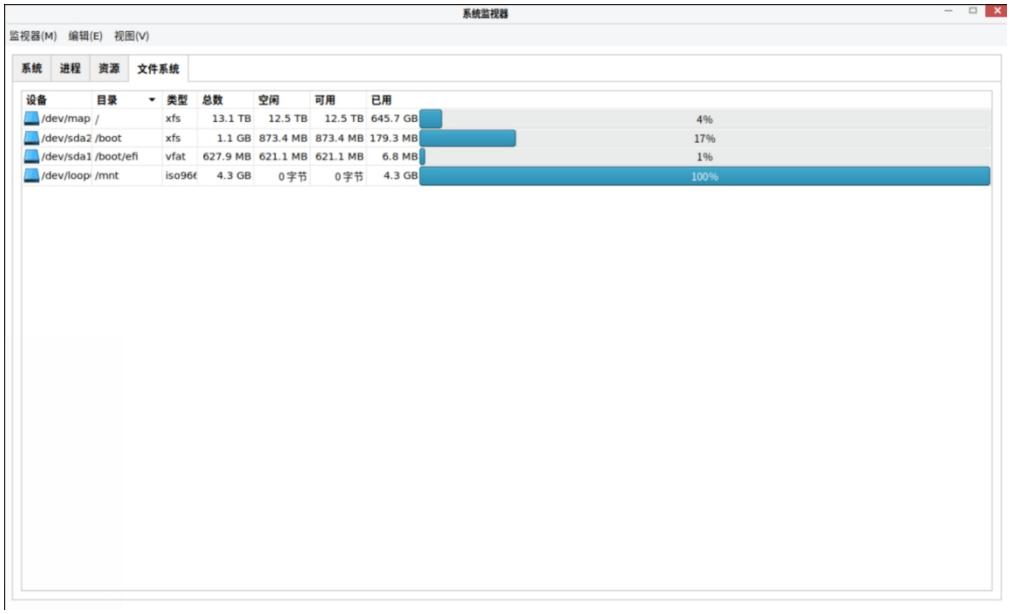


图 7-3 文件系统

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。

7.1.4. 查看硬件信息

7.1.4.1. 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。

7.1.4.2. 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。

7.1.4.3. 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，型号，家族，模式，cpu 高速缓存等等，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。

7.1.5. 检查硬件错误

银河麒麟高级服务器操作系统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。

```
#dnf 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 手册里也有描述。

7.1.6. 使用 Net-SNMP 监控性能

银河麒麟高级服务器操作系统包括 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。

7.1.6.1. 安装 Net-SNMP

Net-SNMP 软件套件可作为银河麒麟高级服务器操作系统 V10 发行版的一

组 RPM 软件包。表 7-2 可用 Net-SNMP 软件包概述了每个包和它们的内容。

表 7-2 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 dnf 命令：

```
dnf 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：

```
#dnf install net-snmp net-snmp-libs net-snmp-utils
```

7.1.6.2. 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。有关在银河麒麟高级服务器操作系统 V10 系统的管理服务的详细信息请参阅 5.1 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。

7.1.6.3. 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。银河麒麟高级服务器操作系统 V10 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 命令测试以下配置是否生效。

```
#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]
```

7.1.6.4. 检索数据性能

银河麒麟高级服务器操作系统的 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 7.1.7.3 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 7-3 可用的 OIDs 总结了在 MIB 下可以获取的不同的 OID。

表 7-3 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::hrSWRu

	n。
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/mibs/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 HOST-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::hrStorageUsed 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::diskIOTable 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 diskIONWrittenX 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）。

包括银河麒麟高级服务器操作系统 V10 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-EXTEND-MIB::nsExtendObjects 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 nsExtendObjects 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。

7.2. 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 7.5.5 使能持续存储。有关期刊详细信息，请参阅 7.5 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 7.2.7 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 7.3Syslogd 日志结构）

7.2.1. 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。

7.2.2. 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）。

7.2.2.1. 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通过任何属性过滤系统日志消息，如 timegenerated 或 syslogtag。您可以将每个指定属性和值比较，这些值和说明参见表 7-4 基于属性的比较操作。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 7-4 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 7-4 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 startswith '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 7.2.3 使用新的配置格式。

7.2.2.2. 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]
```

- “@”符号表示 syslog 消息被转发到使用 UDP 协议的主机；
- “@@”符号表示 syslog 消息被转发到使用 TCP 协议的主机；
- 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 7.2.2.4 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 100MB）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 7.2.6 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。

7.2.2.3. 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 50000 个消息，可以修改数字重新定义。

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（ 7.2.3 使用新的配置格

式）。然而，经典的全局指令仍然为旧格式提供支持。

7.2.2.4. 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。

7.2.3. 使用新的配置格式

银河麒麟高级服务器操作系统安装的 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 7.2.2.1 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。

7.2.3.1. 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。

7.2.3.2. 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。

7.2.4. 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：

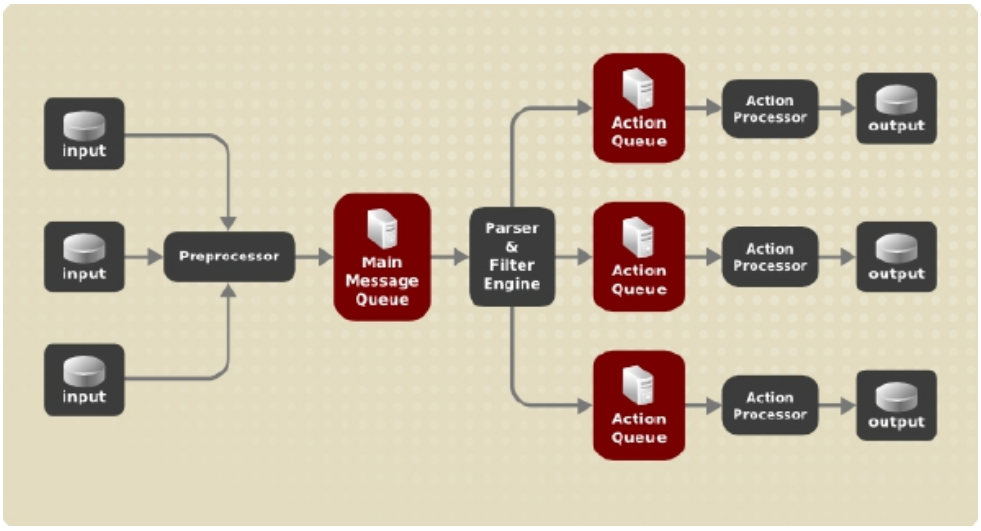


图 7-4Rsyslog 中消息流

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的消息，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。

7.2.4.1. 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 10MB。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。

7.2.4.2. 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时，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
```

如果按上面那样设置，所有队列的数据在 rsyslogd 终止前都会被存储到磁盘。

7.2.4.3. 使用 rsyslog 队列新语法

从 rsyslog 7 开始 rsyslog 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。

7.2.5. 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。银河麒麟高级服务器操作系统 V10 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：

```
#dnf 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-utils 包的一部分提供。如果需要的话，安装包如下：

```
#dnf install policycoreutils-python-utils
```

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：

```
#semanage port -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 身份来执行命令。

7.2.5.1. 在日志服务器上使用模板

rsyslog 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:::s
ecpathreplace%.log")
  template(name="TmplMsg" type="string"
string="/var/log/remote/msg/%HOSTNAME%/%PROGRAMNAME:::s
ecpathreplace%.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")
```

7.2.6. 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 omsnmp，omrelp 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 mmjsonparse 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 rsyslog 提供的模板功能相互协同。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。

7.2.6.1. 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 rsyslog 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。

7.2.6.2. 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。

7.2.6.3. 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。

7.2.7. 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 7.3Syslogd 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 7.3.1 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
```

7.3. 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。结构化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 7.3.2 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。

7.3.1. 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 7.2.7SSyslogd 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。

7.3.2. 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 omfile 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")
```

7.3.3. 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。

7.3.4. 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 7.2.3 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。

7.4. 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。

7.5. 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。

7.5.1. 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。

7.5.2. 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。

7.5.3. 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。

7.5.4. 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 ...
```

7.5.5. 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
```

7.6. 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。

银河麒麟高级服务器操作系统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。

银河麒麟高级服务器操作系统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 7.6.1Cron 和 Anacron 和 7.6.8At 和 Batch）。

银河麒麟高级服务器操作系统 V10 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。

7.6.1. 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。

7.6.2. 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 dnf 命令：

```
dnf 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：

```
#dnf install cronie cronie-anacron
```

要了解如何在银河麒麟高级服务器操作系统中安装新的包,请参见 4.2.4 安装软件包。

7.6.3. 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。要了解更多有关在银河麒麟高级服务器操作系统 V10 中如何管理服务的通用方法，请参见 5.1 使用 systemd 管理系统服务。

7.6.3.1. 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
```

7.6.3.2. 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
```

7.6.3.3. 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。

7.6.4. 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。

7.6.4.1. 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.daily
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）。

7.6.5. 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 7.6.4 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。

7.6.6. 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。

7.6.7. 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 8.7.6 控制对 Cron 的访问中描述的 cron.deny 和 cron.allow 的使用规则一样。

7.6.8. 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。

7.6.8.1. 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 dnf 命令：

```
dnf 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：

```
#dnf install at
```

7.6.8.2. 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。

7.6.8.3. 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+5days 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 7.6.8.5 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 7.6.8.7 控制对 At 和 Batch 的访问。

7.6.8.4. 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 7.6.8.5 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 7.6.8.7 控制对 At 和 Batch 的访问。

7.6.8.5. 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。

7.6.8.6. 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 7-5at 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。

7.6.8.7. 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。

第八章 系统安全

8.1. 安全基础服务

8.1.1. 防火墙

防火墙是系统的主要的安全工具，可以提供基本的安全防护。firewalld 支持网络/防火墙区域(zone)定义网络链接、是接口安全等级的动态防火墙管理工具。它支持 IPv4,IPv6 防火墙设置以及以太网桥接，并且拥有运行时配置和永久配置选项。它也支持允许服务或者应用程序直接添加防火墙规则的接口且可以动态管理防火墙。

8.1.1.1. 主要功能

1) 实现动态管理，对于规则的更改不再需要重新创建整个防火墙；

2) 提供 firewall-cmd 命令行界面进行管理及配置工作；

3) 实现 firewall-config 图形化配置工具；

4) 实现系统全局及用户进程的防火墙规则配置管理；

5) 区域支持。

8.1.1.2. 基本命令

安装命令如下：

```
dnf install firewalld
```

启动命令如下：

```
systemctl enable firewalld.service
systemctl start firewalld.service
```

关闭命令如下：

```
systemctl stop firewalld
systemctl disable firewalld
```

查看状态命令如下：

```
systemctl status firewalld
```

8.1.1.3. 区域管理

通过将网络划分成不同的区域（通常情况下称为 zones），制定出不同区域之间的访问控制策略来控制其间传送的数据流。例如互联网是不可信任的区域，而内部网络是高度信任的区域,以避免安全策略中禁止的一些通信。典型信任的区域包括互联网(一个没有信任的区域)和一个内部网络(一个高信任的区域)。最终目标是提供受控连通性在不同水平的信任区域通过安全政策的运行和连通性模型之间根据最少特权原则。例如：公共 WIFI 网络连接应该不信任，而家庭有线网络连接就应该完全信任。网络安全模型可以在安装、初次启动和首次建立网络连接时选择初始化。该模型描述了主机所联的整个网络环境的可信级别，并定义了新连接的处理方式。在/etc/firewalld/的区域设定是一系列可以被快速执行到网络接口的预设定。有几种不同的初始化区域：

drop（丢弃）

任何接收的网络数据包都被丢弃，没有任何回复。仅能有发送出去的网络连接。

block（限制）

任何接收的网络连接都被 IPv4 的 icmp-host-prohibited 信息和 IPv6 的 icmp6-adm-prohibited 信息所拒绝。

public（公共）

在公共区域内使用，不能相信网络内的其他计算机，只能接收经过选取的连接。

external（外部）

特别是为路由器启用了伪装功能的外部网。您不能信任来自网络的其他计算机，只能接收经过选择的连接。

dmz（非军事区）

用于您的非军事区内的电脑，此区域内可公开访问，可以有限地进入您的内部网络，仅仅接收经过选择的连接。

work（工作）

用于工作区。您可以基本相信网络内的其他电脑不会危害您的电脑。仅仅接收经过选择的连接。

home（家庭）

用于家庭网络。您可以基本信任网络内的其他计算机不会危害您的计算机。仅仅接收经过选择的连接。

internal（内部）

用于内部网络。您可以基本上信任网络内的其他计算机不会威胁您的计算机。仅仅接受经过选择的连接。

trusted（信任）

可接受所有的网络连接。

说明：firewalld 的缺省区域是 public。

显示支持的区域列表

```
firewall-cmd --get-zones
```

设置为家庭区域

```
firewall-cmd --set-default-zone=home
```

查看当前的区域

```
firewall-cmd --get-active-zones
```

设置当前的区域的接口

```
firewall-cmd --get-zone-of-interface=enp03s
```

显示所有公共区域（public）

```
firewall-cmd --zone=public --list-all
```

临时修改网络接口 enp0s3 为内部区域（internal）

```
firewall-cmd --zone=internal --change-interface=enp03s
```

永久修改网络接口 enp0s3 为内部区域（internal）

```
firewall-cmd --permanent --zone=internal --change-interface=enp03s
```

8.1.1.4. 服务管理

amanda、ftp、samba 和 tftp 等重要的服务已被 firewalld 提供相应的服务，可以使用命令查看：

```
firewall-cmd --get-services
```

显示当前服务

```
firewall-cmd --list-services
```

添加 http 服务到内部区域（internal）

```
    firewall-cmd --permanent --zone=internal
--add-service=http
    firewall-cmd --reload
```

将一个服务加入到分区

要把一个服务加入到分区，例如允许 SMTP 接入工作区：

```
    firewall-cmd --zone=work --add-service=smtp
    firewall-cmd --reload
```

要从分区移除服务，比如从工作区移除 SMTP：

```
    firewall-cmd --zone=work --remove-service=smtp
    firewall-cmd --reload
```

8.1.1.5. 端口管理

打开端口

打开 443/tcp 端口在内部区域（internal）：

```
firewall-cmd --zone=internal --add-port=443/tcp
firewall-cmd --reload
```

端口转发

```
firewall-cmd --zone=external --add-masquerade
firewall-cmd --zone=external
--add-forward-port=port=22:proto=tcp:toport=3777
```

上面的两个命令的意思是，首先启用伪装（masquerade），然后把外部区域（external）的 22 端口转发到 3777。

8.1.1.6. 直接接口

firewalld 有一个被称为“direct interface”（直接接口），它可以直接通过 iptables、ip6tables 和 ebtables 的规则。它适用于应用程序，而不是用户。

firewalld 保持对所增加项目的追踪，所以它还能质询 firewalld 和发现由使用直接端口模式的程序造成的更改。直接端口由增加--direct 选项使用。直接端口模式适用于服务或者程序，以便在运行时间内增加特定的防火墙规则。这些规则不是永久性的。例如添加端口 tcp 9000 端口。

```
firewall-cmd --direct --add-rule ipv4 filter INPUT 0 -p tcp --dport 9000 -j ACCEPT
firewall-cmd --reload
```

8.1.1.7. 富规则（Rich Language）

通过“rich language”语法，可以用比直接接口方式更易理解的方法建立复杂防火墙规则。此外还能永久保留设置。这种语言可以用来配置分区，也仍然支持现行的配置方式。所有命令都必须以 root 用户身份运行。增加一项规则的命令格式如下：

```
#firewall-cmd [--zone=zone] --add-rich-rule='rule' [--timeout 9=seconds]
移除一项规则：
firewall-cmd [--zone=zone] --remove-rich-rule='rule'
检查一项规则是否存在：
firewall-cmd [--zone=zone] --query-rich-rule='rule'
```

一个具体例子，假设在一个 IP 地址（192.168.0.0）的服务器配置防火墙允许如下服务 http,https,vnc-server,PostgreSQL。

```
firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="http" accept'
firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="http" accept' --permanent
firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="https" accept'
```

```
firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="https" accept' --permanent
  firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="vnc-server" accept'
  firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="vnc-server" accept' --permanent
  firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="postgresql" accept'
  firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24" service name="postgresql" accept' --permanent
firewall-cmd --reload
```

8.1.1.8. 创建服务

首先需要建立的服务是 RTMP（RTMP 是 Real Time Messaging Protocol（实时消息传输协议）的首字母缩写。该协议基于 TCP）端口号 1935。在/etc/firewalld/services/目录中，利用现有的配置文件如 nfs.xml 作为模板。

```
cd /etc/firewalld/services/
```

说明：该目录中存放的是定义好的网络服务和端口参数，只用于参考，不能修改。这个目录中只定义了一部分通用网络服务。在该目录中没有定义的网络服务，也不必再增加相关 xml 定义，后续通过管理命令可以直接增加。

```
cp /usr/lib/firewalld/services/nfs.xml /etc/firewalld/services/
```

说明：从上面目录中将需要使用的服务的 xml 文件拷至这个目录中，如果端口有变化则可以修改文件中的数值。

```
#cd /etc/firewalld/services/
```

```
下面修改 nfs.xml 为 rtmp.xml

#mv nfs.xml rtmp.xml
```

下面使用 vi 编辑器修改 rtmp.xml 文件为如下内容：

```
<?xml version="1.0" encoding="utf-8"?>
 <service>
 <short>rtmp </short>
 <description>RTMP Stream </description>
 <port protocol="tcp" port="1935"/>
 </service>
```

每一个服务定义都需要一个简短的名字、描述和端口网络用于指定需要使用的协议、端口和模块名。然后把此服务加入防火墙规则中。

```
systemctl restart firewalld.service
firewall-cmd --add-service=rtmp
firewall-cmd --add-service=rtmp --permanent
firewall-cmd --reload
```

8.1.1.9. firewall-config

firewall-config 支持防火墙的所有特性。管理员可以用它来改变系统或用户策略。通过 firewall-config 用户可以配置防火墙允许通过的服务、端口、伪装、端口转发、和 ICMP 过滤器和调整 zone（区域）设置等功能以使防火墙设置更加的自由、安全和强健。

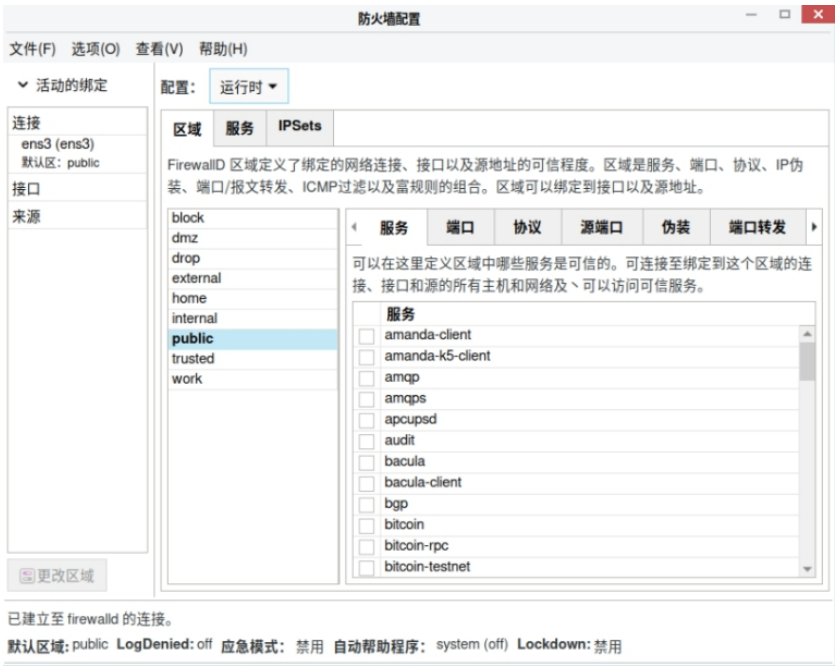


图 8-1 防火墙配置

firewall-config 工作界面分成三个部分：上面是主菜单，中间是配置选项卡。下面是区域、服务、ICMP 端口、白名单等设置选项卡。

说明：在左下方角落寻找“连接”字符，这标志着 firewall-config 工具已经连接到用户区后台程序 firewalld。注意，ICMP 类型、直接配置 （Direct Configuration）和锁定白名单 （Lockdown Whitlist）标签只在从查看下拉菜单中选择之后才能看见。

1）firewall-config 主菜单

firewall-config 主菜单包括四个选项：文件，选项，查看，帮助。其中选项子菜单是最主要的，它包括几个部分：

重载防火墙：重载防火墙规则。例如所有现在运行的配置规则如果没有在永久配置中操作，那么系统重载后会丢失。

更改连接区域：更改网络连接的默认区域。

改变默认区域：更改网络连接的所属区域和接口。

应急模式：应急模式意味着丢弃所有的数据包。

锁定：锁定可以对防火墙配置进行加锁，只允许白名单上的应用程序进行改动。锁定特性为 firewalld 增加了锁定本地应用或者服务配置的简单配置方式。它是一种轻量级的应用程序策略。

2）配置选项卡

firewall-config 配置选项卡包括：运行时和永久。

运行时：运行时配置为当前使用的配置规则。运行时配置并非永久有效，在重新加载时可以被恢复，而系统或者服务重启、停止时，这些选项将会丢失。

永久：永久配置规则在系统或者服务重启的时候使用。永久配置存储在配置文件种，每次机器重启或者服务重启、重新加载时将自动恢复。

3）区域选项卡

区域选项卡是一个主要设置界面：

网络或者防火墙区域定义了连接的可信程度。firewalld 提供了几种预定义的区域。这里的区域是服务、端口、协议、伪装、ICMP 过滤等组合的意思。区域可以绑定到接口和源地址。服务子选项卡定义哪些区域的服务是可信的。可信的服务可以绑定该区的任意连接、接口、和源地址；

i. 服务选项卡

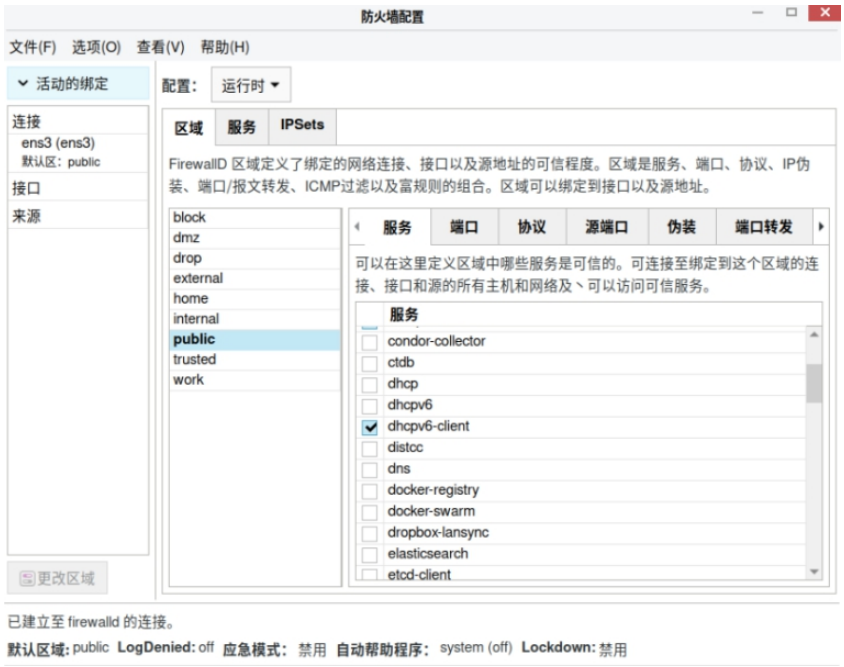


图 8-2 服务选项卡

ii. 端口子选项卡

端口子选项卡用来设置允许主机或者网络访问的端口范围。

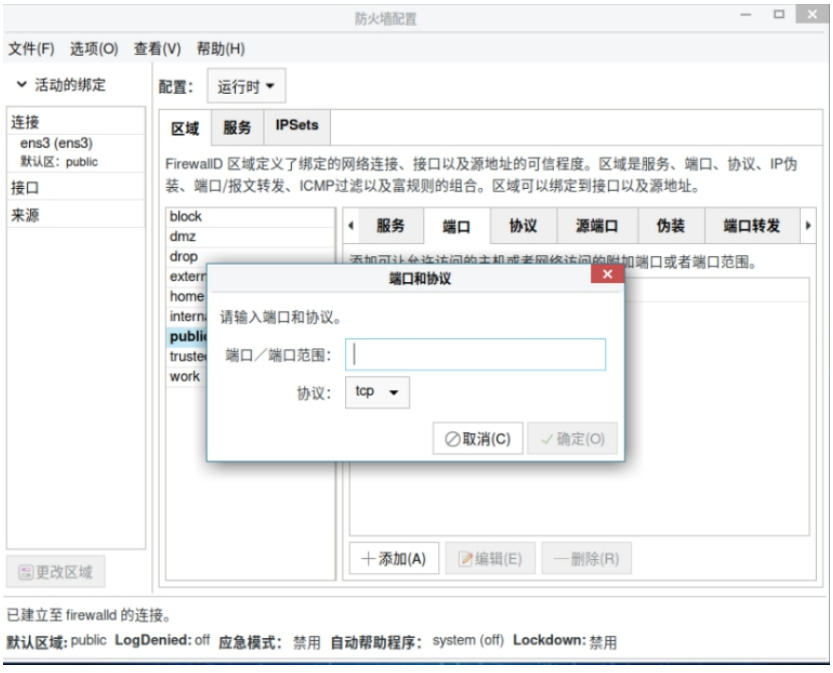


图 8-3 端口子选项卡

要允许流量通过防火墙到达某个端口，则启动 firewall-config 并选择您想

更改设定的网络区域。选择端口图标并点击右边的添加按钮，Port and Protocol 就打开了。

输入端口数量或者端口号范围，获得许可。从下拉菜单中选择 tcp 或者 udp。

iii. 伪装子选项卡

伪装子选项卡用来把私有网络地址可以被映射到公开的 IP 地址

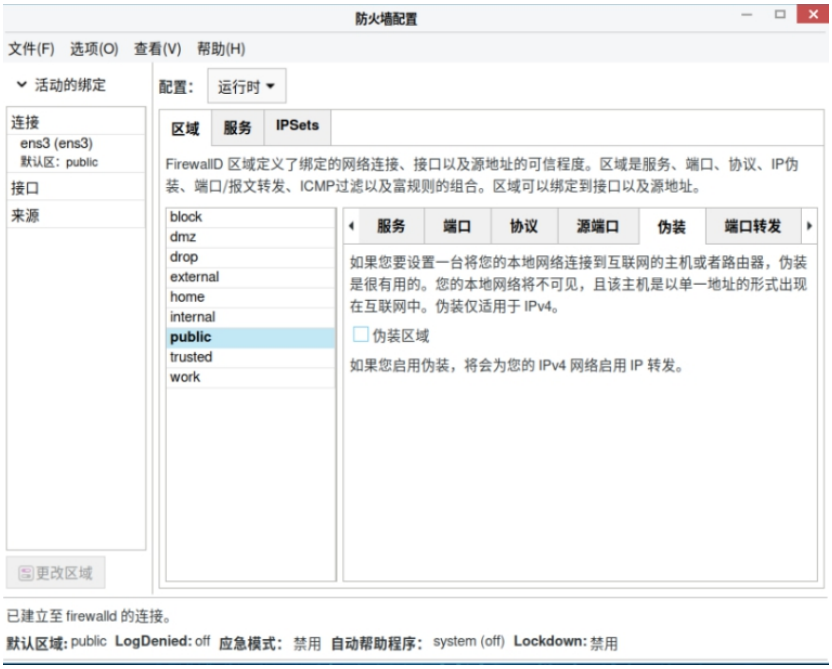


图 8-4 伪装子选项卡

要将 IPv4 地址转换为一个单一的外部地址，则启动 firewall-config 工具并选择需要转换地址的网络区域。选择 伪装标签和复选框以便把 IPv4 地址转换成一个单一的地址。

iv. 端口转发子选项卡

端口转发可以映射到另一个端口以及/或者其他主机;

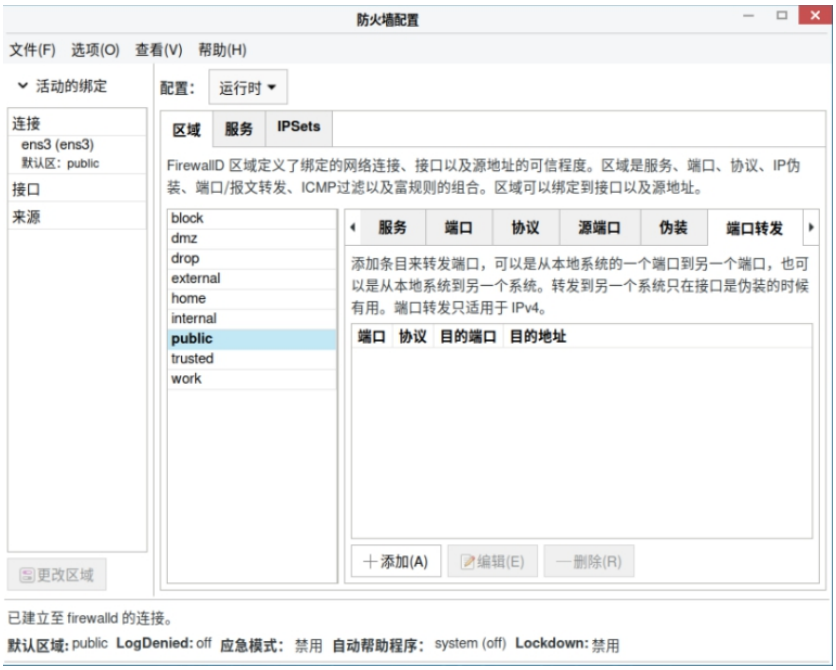


图 8-5 端口转发子选项卡

为一个特定端口转发入站网络流量或“packets”到一个内部地址或者替代端口，首先激活伪装 IP 地址，然后选择端口转发标签。在窗口靠上部分选择入站流量协议和端口或者端口范围。靠下部分是用于设置目的端口细节的。

要转发流量到一个本地端口即同一系统上的端口，需选择本地转发复选框，输入要转发的流量的本地端口或者端口值范围。要转发流量到其他的 IPv4 地址，则选择转发到另一个端口复选框，输入目的地 IP 地址和端口或者端口范围。如果端口位置空缺则默认发送到同一个端口。点击确定按钮执行更改。

v. ICMP 过滤器子选项卡

ICMP 过滤器可以选择 Internet 控制报文协议的报文。这些报文可以是信息请求亦可是对信息请求或错误条件创建的响应；

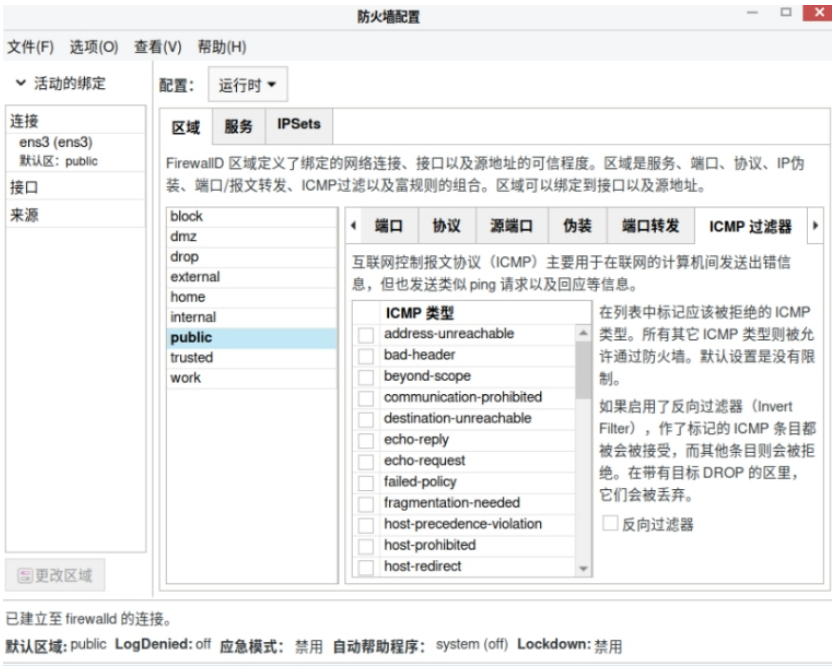


图 8-6ICMP 过滤器子选项卡

要使用或者禁用一个 ICMP 过滤，则启动 firewall-config 工具并选择要过滤其信息的网络区域。选择 ICMP Filter 图标并选择每种您需要过滤的 ICMP 信息类型的复选框。清除复选框以禁用过滤。这种设定是单向的，默认允许全部。

vi. 直接配置选项卡

直接配置选项卡包括三个子选项卡：链、规则和穿通。说明以下这个选项卡主要用于服务或者应用程序增加特定的防火墙规则。这些规则并非永久有效，并且在收到 firewalld 通过 D-Bus 传递的启动、重启、重载信号后需要重新应用。

图 8-7 直接配置选项卡

4) 改变防火墙设置

要立刻改变现在的防火墙设置，须确定当前视图设定在运行时或者从下拉菜单中选择永久（Permanent），编辑下次启动系统或者防火墙重新加载时执行的设定。

在运行时（Runtime）模式下更改防火墙的设定时，一旦您启动或者清除连接服务器的复选框，选择立即生效。在 Permanent 模式下更改防火墙的设定，仅仅在重新加载防火墙或者系统重启之后生效。可以使用文件菜单下的重新加载图标，或者点击选项菜单，选择重新加载防火墙。

图 8-8 改变防火墙设置

5) 修改默认分区

要设定一个将要被分配新接口的分区作为默认值，则启动 firewall-config，从菜单栏选择选项卡，由下拉菜单中选择修改默认区域，从给出的列表中选择您需要用的分区作为默认分区，点击确定按钮即可。

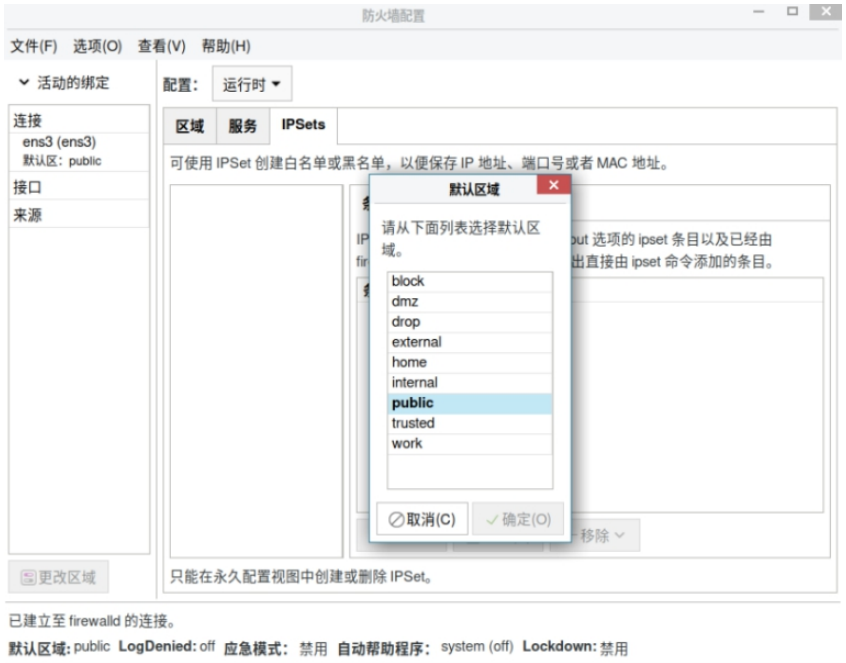


图 8-9 修改默认分区

8.1.2. 审计管理(audit)

8.1.2.1. 简介

audit 工具可以将审计记录写入日志文件。包括记录系统调用和文件访问。管理员可以检查这些日志，确定是否存在安全漏洞（如多次失败的登录尝试，或者用户对系统文件失败访问）。

8.1.2.2. 架构及原理分析

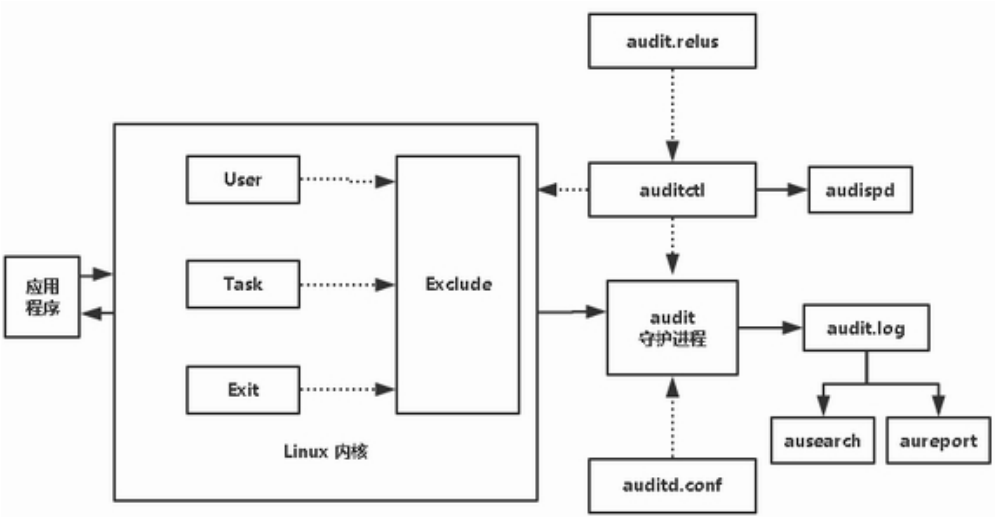


图 8-10 架构示意图

说明：实线代表数据流，虚线代表组件之间的控制关系。包括有两大部分：中间的是内核中的几种系统调用（user,task,exit,exclude），右侧是一系列应用程序（auditd、audispd、auditctl、autrace、ausearch 和 aureport 等）。Linux 内核中的几种系统调用是：

User:记录用户空间中产生的事件；它的作用是过滤消息的，内核传递给审计后台进程之前先查询它。

Task：跟踪应用程序的子进程（fork）；当一个任务被创建时，也就是父进程通过 fork 和克隆创建子进程时记录该事件；

Exit：当一个系统调用结束时判断是否记录该调用；

Exclude：删除不合格事件；Exclude 是用来过滤消息的，也就是不想看到的消息可以在这里写规则进行过滤。

audit 是内核中的一个模块，内核的运行情况都会在 audit 中记录，这个记录的规则是由超级用户来设置的。内核的 audit 模块是由应用层的一个应用程序 auditd 来控制的。audit 产生的数据都会传送到 auditd 中，然后再由 auditd 进行其它操作。auditd.conf 是 auditd 的配置文件。audit.rules 是 audit 的规则文件，确定 audit 的日志中记录哪些操作。它通过一个对 audit 进行控制的应用程序 auditctl 进行操作。root 用户也可以直接调用 auditctl 进行操作。auditd 收到的数据后会有两个去处。默认的是将日志保存在 audit.log 文件中，默认路径/var/log/audit/audit.log。另一个通过 audispd 将日志进行分发。

要使用安全审计系统可采用下面的步骤：首先安装软件包。然后设置配置文件、配置常用命令，添加审计规则，然后启用 audit 守护进程并开始进行日志记录，最后通过生成审计报表和搜索日志来周期性地分析数据。

用户空间审计系统由 auditd、audispd、auditctl、autrace、ausearch 和 aureport 等应用程序组成。下面依次说明：

auditctl:即时控制审计守护进程的行为的工具，如添加规则等。

auditd：audit 守护进程负责把内核产生的信息写入到硬盘上，这些信息由应用程序和系统活动触发产生。用户空间审计系统通过 auditd 后台进程接收内

核审计系统传送来的审计信息，将信息写入到/var/log/audit/audit.log。

aureport:查看和生成审计报告的工具。

ausearch:查找审计事件的工具

auditspd:转发事件通知给其他应用程序，而不是写入到审计日志文件中。

autrace:一个用于跟踪进程的命令。类似于 strace，跟踪某一个进程，并将跟踪的结果写入日志文件之中。

8.1.2.3. 安装及配置

1) 使用 dnf 工具安装软件包

```
dnf install audit -y
```

2) audit 配置文件

audit 安装后会生成 2 个配置文件:/etc/audit/auditd.conf 和/etc/audit/audit.rules。/etc/audit/auditd.conf 是守护程序的默认配置文件。/etc/audit/audit.rules 是记录审计规则的文件。首次安装 audit 后,审计规则文件是空的。

/etc/audit/auditd.conf 守护程序的默认配置文件是其关键文件。

下面简单设置一个/etc/audit/auditd.conf

```
#vi /etc/audit/auditd.conf
#设置日志文件
log_file = /var/log/audit/audit.log
#设置日志文件轮询的数目，它是 0~99 之间的数。如果设置为小于 2，则不会循环日志。如果没有设置 num_logs 值，它就默认为 0，意味着从来不循
```

```
环日志文件
    num_logs = 5
    #设置日志文件是否使用主机名称
    name_format = NONE
    #设置日志文件大小，以兆字节表示的最大日志文件容量。当达到这个容量
时，会执行 max_log_file_action 指定的动作
    max_log_file = 6
    #设置日志文件到达最大值后的动作，这里选择 ROTATE（轮询）
    max_log_file_action = ROTATE
```

8.1.2.4. auditctl 命令简介

auditctl 命令格式如下：

```
auditctl [选项] filter,action -S syscall -F condition -k label
```

表 8-1 auditctl 命令选项

项目	可选参数	说明
filter	user,exit,task,exclude	filter 详细说明哪个内核规则匹配过滤器应用在事件中。过滤器:task、exit、user 以及 exclude
action	always,never	是否审核事件（always 表示是）（never 表示否）
syscall	all,2,open 等	所有的系统调用都可以在/usr/include/asm/unistd_64.h 文件中找到。

condition	euid=0,arch=b64	详细说明其他选项，进一步修改规则来与特定架构、组 ID、进程 ID 和其他内容为基础的事件相匹配
label	任意文字	标记审核事件并检索日志

-S 表示系统调用号或名字；

-F 表示规则域；

-k 表示设置审计规则上的过滤关键。

8.1.2.5. audit 审计规则

audit 审计规则分成三个部分：

1) 控制规则：这些规则用于更改审计系统本身的配置和设置。

控制规则可以在/etc/audit/audit.rules 中设置。主要包括:

-D#删除所有当前装载的审核规则#

-b 8192#在内核中设定最大数量的已存在的审核缓冲区为 8Mb#

-e 2 #锁定审核配置#

2) 文件系统规则：这些是文件或目录监视。使用这些规则，我们可以审核对特定文件或目录的任何类型的访问。

可以通过 auditctl 命令设置。监控文件系统行为（依靠文件、目录的权限属性来识别）

规则格式：

-w 路径；

-p 权限；

-k 关键字；

其中-p 权限的动作分为四种：

r — 读取文件或者目录；

w — 写入文件或者目录；

x — 运行文件或者目录；

a — 改变在文件或者目录中的属性。

例如要监控/etc/passwd 文件的修改行为，可以使用这个命令：

```
auditctl -w /etc/passwd -p wa
```

也可以自己将上述内容加入到文件/etc/audit/rules.d/audit.rules 中即可实现对该文件的监视。

下面审核规则记录了每次读取或者修改/etc/hosts 文件的尝试

```
auditctl -w /etc/hosts -p wa -k hosts_change
```

3) 系统调用规则：这些规则用于监视由任何进程或特定用户进行的系统调用。

监控系统调用可能会引起高负荷的日志活动，这会让内核承受更大的负荷。所以要慎重衡量哪些系统调用需要放到audit.rules中。如果审计的是目录的话，只能对该目录本身的属性进行审计。如果想审计下面的文件，需要一一列出。

系统调用的监控：

-a 添加一条系统调用监控规则；

-S 显示需要监测的系统调用的名称。

显示规则和删除规则：

-D 删除所有规则；

-d 删除一条规则和-a 对应；

-w 写入文件或者目录；

-W 删除一条规则和-w 对应；

-l 列出所有规则。

举例:定义记录有哪些文件，特定用户（UID 为 10001）访问和标签的日志条目的规则：

```
auditctl -a always,exit -F arch=b64 -F auid=10001 -S open -k userfile
```

说明：userfile 是用户自己设置的一个规则名字，以上的都设置完毕了，就可以生成报告了。另外通过 auditctl 命令添加的规则不是永久有效的。为了让他们在重新启动后有效的，可以将其添加到文件 /etc/audit/rules.d/audit.rules 中。

8.1.2.6. 守护进程

启动：

```
systemctl start auditd
```

自动启动：

```
systemctl enable auditd
```

停止：

```
service auditd stop
```

重启：

```
service auditd restart
```

在/var/log/audit/目录中旋转日志文件：

```
service auditd rotate
```

推迟审核事件日志之后重新开始，例如存在没有足够的磁盘分区空间来保存。审核日志文件情况：

```
service auditd resume
```

显示运行状态：

```
service auditd status
```

列出所有活动的规则和观察器：

```
auditctl -l
```

配置 auditd 的日志文件到远程主机。

假设 audit 服务器名称是 kylin1,ip 地址 192.168.0.1;audit 客户端名称是 kylin2，ip 地址 192.168.0.2。

修改服务器端配置文件：

```
vi /etc/audit/auditd.conf
#设置监听端口为 60
tcp_listen_port =60

重启服务
#service auditd restart
```

客户端安装相关软件包并且配置文件：

```
#dnf -y install audispd-plugins
修改配置文件
```

```
#vi /etc/audit/plugins.d/au-remote.conf
#把审核日志发送到日志服务器
active =yes

#vi /etc/audit/audisp-remote.conf
remote_server =kylin1

#设置监听端口为 60
port = 60

#vi /etc/audit/auditd.conf
log_format =NOLOG

然后重启服务
#service auditd restart
```

8.1.2.7. aureport

要生成审计消息的报表，可使用 aureport 命令。如果执行 aureport 时没有使用任何选项，则会显示如汇总报表。下面是几个例子:

生成一段特定时间内的报告：

```
aureport -ts 8:00 -te 17:30 -f -i
```

生成所有用户失败事件的总结报告：

```
aureport -u --failed --summary -i
```

生成系统调用事件报告：

```
aureport -s -i --summary
```

8.1.2.8. ausearch

使用 ausearch，您可以过滤和搜索事件类型。它还可以通过将数值转换

为更加直观的值(如系统调用或用户名)来解释事件。以 root 用户执行 ausearch 命令，当显示结果时，每个记录用 4 条虚线组成的一行隔开，每个记录前均显示时间标记。

示例：ausearch -f /etc/passwd

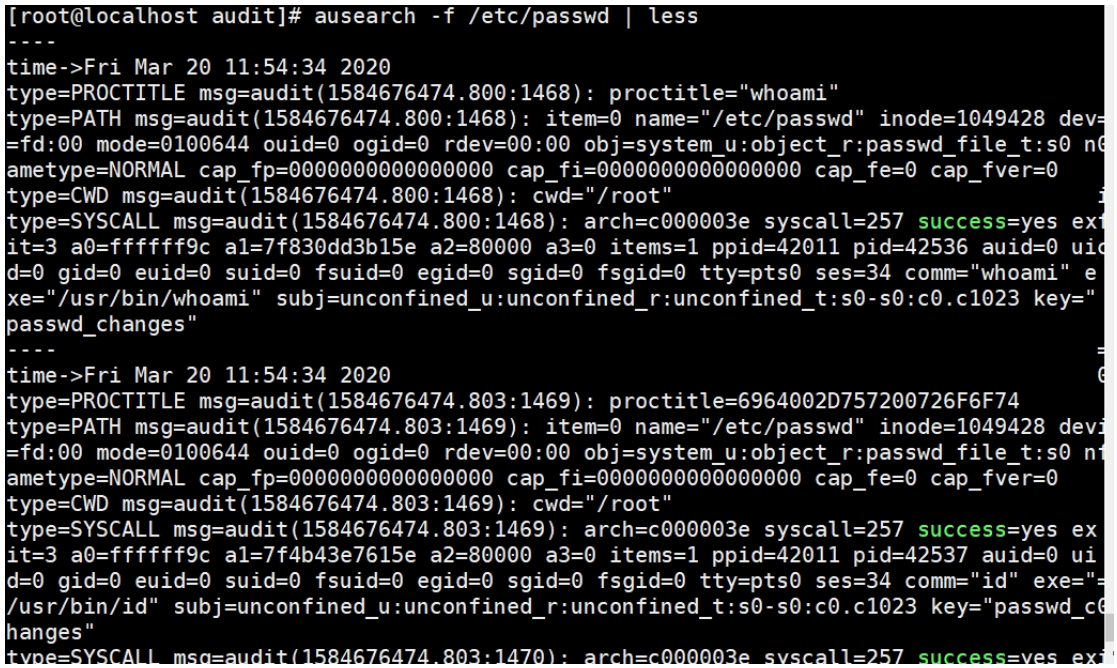


图 8-11 输出结果

输出结果介绍:

time:审计时间；

name:审计对象；

cwd:当前路径；

syscall:相关的系统调用；

auid:审计用户 ID；

uid 和 gid:访问文件的用户 ID 和用户组 ID；

comm:用户访问文件的命令；

exe:上面命令的可执行文件路径。

下面是几个例子：

搜索系统登录失败,使用如下命令：

```
ausearch --message USER_LOGIN --success no –interpret
```

搜索所有的账户，群组，角色变更，使用以下命令：

```
ausearch -m ADD_USER -m DEL_USER -m ADD_GROUP -m USER_CHAUTHTOK -m DEL_GROUP -m CHGRP_ID -m ROLE_ASSIGN -m ROLE_REMOVE -i
```

搜索从制定时间段的失败的系统调用，使用以下命令：

```
ausearch --start yesterday --end now -m SYSCALL -sv no -i
```

使用关键字（key）搜索审计事件记录。

auditd 生成的日志文件内容比较多，如果没有设置关键字，要查找搜索审计事件记录,使用关键字（key）可以提高效率。

首先使用 auditctl 命令建立一个规则：

```
auditctl -w /etc/passwd -p rwa -k passwd_changes
```

上面这个规则的意思是记录访问或修改/etc/passwd 用户帐户数据库的任何尝试。一旦有人修改了数据库文件，可以使用如下关键字（key）搜索审计事件记录。

使用关键字（key）搜索审计事件记录：

```
ausearch -k passwd_changes | less
```

上面的命令显示访问或修改/etc/passwd 文件的日志信息，其他信息忽略。

8.2. 安全增强组件

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银河麒麟服务器操作系统中检索这类信息。

8.2.1. KYSEC 安全机制

银河麒麟高级服务器操作系统 V10 结合国产操作系统特点和现有国内外操作系统的安全机制，提出了基于标记的软件执行控制机制，实现对系统应用程序标记识别和执行约束，确保应用来源的可靠性和应用本身的完整性。执行控制机制控制文件执行、模块加载和共享库使用，分为系统文件、第三方应用程序，其中只允许具有合法标记的文件执行，任何网络下载、拷贝等外来软件均被禁止执行。

8.2.1.1. 安全状态设置

KYSEC 共有两种状态；强制模式（normal）、软模式（softmode）。强制模式下不能执行非法应用程序，即禁止用户执行没有合法标记的应用程序，而软模式下，只是记录非法应用程序的执行行为，不禁止用户操作。系统默认 KYSEC 安全状态为关闭状态，且执行控制、文件保护、内核保护状态都处于关闭状态，三权分立关闭。

查看 kysec 安全状态：

```
#getstatus
KySec status: Normal
```

```
exec control: on
file protect: on
kmod protect: on
three admin : off
```

设置 kysec 安全状态为软模式：

```
#sudo setstatus softmode
```

设置 kysec 安全状态为强制模式：

```
#sudo setstatus normal
```

8.2.1.2. 执行控制

银河麒麟高级服务器操作系统 V10 添加了对文件执行、模块加载和共享库使用的控制，分为系统文件、第三方应用程序，其中只允许具有合法标记的文件执行，任何网络下载、拷贝等外来软件均被禁止执行。本来可运行的可执行文件在删除、修改、拷贝、移动、重命名等操作后，会被禁止执行。

（1）可执行文件的执行控制

如下情况的/tmp/ls 则执行失败。

```
#cp /bin/ls /tmp
#/tmp/ls
```

但管理员设置/tmp/ls 的 kysec 标记后，用户则可执行/tmp/ls 程序：

```
#sudo kysec_set -n exectl -v original /tmp/ls
#/tmp/ls
```

篡改/tmp/ls 程序内容后，该程序不能再执行：

```
#echo “ ” >> /tmp/ls
#/tmp/ls
```

（2）可执行脚本的执行控制

创建脚本文件/tmp/test.sh 并添加执行权限,内容如下：

```
#!/bin/bash
echo “test”
#chmod +x /tmp/test.sh
```

但用户使用如下命令，都无法执行程序：

```
#/tmp/test.sh
#bash /tmp/test.sh
```

管理员用户设置该文件的 kysec 标记后，程序就能正常执行。

```
#sudo kysec_set -n exectl -v original /tmp/test.sh
```

（3）内核模块的执行控制

用户将可加载的内核模块复制到主目录，再尝试加载拷贝的内核模块，因为 KYSEC 功能，则加载失败：

```
#cp /lib/modules/`uname -r`/kernel/net/netfilter/nf_conntrack_ftp.ko ~
#sudo insmod ~/nf_conntrack_ftp.ko
```

修改拷贝的内核模块标记，则可以加载成功。

```
#sudo kysec_set -n exectl -v original ~/nf_conntrack_ftp.ko
#sudo insmod ~/nf_conntrack_ftp.ko
```

8.2.1.3. 文件保护功能

文件保护功能开启时，受保护的文件不能被修改、重命名、删除。这样能保护系统重要的配置文件不被误操作修改。

给文件设置文件保护的标记：

```
#sudo kysec_set -n protect -v readonly /tmp/testfile
```

给文件移除文件保护的标记：

```
#sudo kysec_set -xn protect /tmp/testfile
```

给文件/tmp/testfile 设置文件保护的标记后，则禁止修改,重命名,删除/tmp/testfile 文件。如下操作则会提示失败。

```
#echo “test1” >> /tmp/testfile
#mv /tmp/testfile /tmp/abc
#rm -f /tmp/testfile
```

8.2.2. 数据隔离保护机制

银河麒麟高级服务器操作系统 V10 实现用户私有数据的机密性隔离存储；并基于该保护机制，研制了文件保护箱功能，确保每个用户的文件保护箱相互隔离，即使非法用户获得管理员权限也无法获取其他用户的私有数据；同时也提供文件保护箱的数据共享机制。

- 隔离隐藏，用户私有数据仅对自己或共享的系统用户可见，对没有授权的用户实现隐藏隔离；
- 数据共享，每个用户均可对自己的数据进行共享配置，可设置赋予其它用户读、写权限，实现数据共享。

8.2.2.1. 隔离隐藏

数据隔离保护机制的操作目录为/box。用户使用 boxadm 创建一个麒麟文件保护箱，其中的文件、目录等就不能被别的用户访问。

test 用户创建一个文件保护箱 testbox，并新建一个文件：

```
#boxadm -c testbox
#touch /box/test/testbox/testfile
```

test 用户查看/box/test 的目录及文件：

```
#ls -l /box/test
testbox
#ls -l /box/test/testbox
testfile
```

但 root 用户查看/box 下的子目录，如：

（1）ls -l /box/test 命令输出中没有/box/test/testbox，提示“总用量 0”；

（2）ls -ld /box/test/testbox 命令提示”No such file or directory”；

（3）ls -l /box/test/testbox/testfile 命令提示” No such file or directory”

```
#ls -l /box/test
#ls -ld /box/test/testbox
#ls -l /box/test/testbox/testfile
```

8.2.2.2. 数据共享

用户使用 boxadm 设置文件保护箱中的目录共享权限，如只读共享权限、

读写共享权限，被授权的用户就能访问这个目录。

test 用户设置 root 对/box/test/testbox 目录的只读共享权限后，root 用户才能访问/box/test/testbox 目录：

```
#boxadm -s -u root -p rx testbox
```

root 用户执行如下命令，则可以看到/box/test 目录下存在 testbox 目录、/box/test/testbox 目录下存在 testfile 文件。

```
#ls -l /box/test/
#ls -l /box/test/testbox
```

8.2.2.3. boxadm 的使用方法

test 用户创建一个麒麟文件保护箱 testbox，会生成目录/box/test/testbox：

```
#boxadm -c testbox
```

设置 root 对/box/test/testbox 目录的只读共享权限后，root 用户才能访问/box/test/testbox 目录：

```
#boxadm -s -u root -p rx testbox
```

设置 root 对/box/test/testbox 目录的读写共享权限后，root 用户才能读写/box/test/testbox 目录：

```
#boxadm -s -u root -p rwx testbox
```

查看/box/test/testbox 目录的 Box 信息：

```
#boxadm -i testbox
```

```
Access:user:root:rwx
```

移除 root 用户对/box/test/testbox 目录的共享权限：

```
#boxadm -s -u root -X testbox
```

删除/box/test/testbox 目录：

```
#boxadm -r testbox
```

8.2.3. 强制访问控制

在计算机安全领域指一种由操作系统约束的访问控制，目标是限制主体或发起者访问或对对象或目标执行某种操作的能力。在实践中，主体通常是一个进程或线程，对象可能是文件、目录、TCP/UDP 端口、共享内存段、I/O 设备等。主体和对象各自具有一组安全属性。每当主体尝试访问对象时，都会由操作系统内核强制施行授权规则——检查安全属性并决定是否可进行访问。任何主体对任何客体的任何操作都将根据一组授权规则（也称策略）进行测试，决定操作是否允许。

主体和客体：主体指的是访问者(一般为一个进程)，客体指的是被访问的资源。根据资源的类型不同，在策略配置语言中被分为不同的客体，如 dir、file、process 等。

安全上下文：指标记在所有事物上的扩展属性，包括 SELinux 用户、角色、域或类型，如果使用 MLS(多级安全，以下会有介绍)策略还包括安全级。

类型和域：是分配给一个对象并决定谁可以访问这个对象，域对应进程，类型对应其他对象。

域转换：进程以给定的进程类型运行的能力称为域转换，需满足三个转换条

件：一个进程新的域类型有对一个可执行文件类型的 entrypoint 访问权限，进程的当前（或以前）域类型对入口点文件类型拥有 execute 访问权限，进程的当前域类型对新的域类型拥有 transition 访问权限。

角色：和一些域相关联，决定哪些域可以使用。

安全策略用户：和标准 linux 用户对应，影响哪个域可以进入。

8.2.3.1. SELinux 基础策略

银河麒麟高级服务器操作系统 V10 根据系统安全需求，定制了系统图形登录功能策略、三权分立功能策略、审计服务策略、执行控制功能策略、白名单功能策略、kvm/lxc 等系统功能策略、系统使用修订桌面常用工具策略、系统启动时自动标记脚本功能。根据不同应用场景，定制了 ukmls、ukmcs 和 target 三套 SE 基础策略。

8.2.3.2. 安全策略模式切换

系统在正常启动后就进入了允许模式。允许模式是为了方便我们服务器配置，即使我们的安全策略没有允许这样的操作仍可以执行，但会被审计下来；强制模式是只要没有这样的规则操作就不被允许。我们的安全服务器系统在普通模式下使用的就是允许模式的策略，在默认模式下则使用强制模式的策略来加固我们服务器的安全。

（1）当前安全策略模式查询

我们提供了一些工具命令可以查询当前的安全策略模式，如 /usr/sbin/getenforce 和/usr/sbin/sestatus。所有用户都可以通过这些命令进行查询。

getenforce 命令输出若为 Permissive 则表示当前处于允许模式，若为 Enforcing 则表示当前处于强制模式。

sestatus 命令的输出可能为以下内容：

```
#sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             ukmcs
Current mode:                   permissive
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      30
```

它不仅反应了当前安全策略的模式，同时也可以得到其他安全策略的信息：安全策略开启状态、安全策略文件系统挂载位置、配置文件中的默认模式、安全策略版本和安全策略目录名称。

（2）安全策略模式修改

为了安全考虑所以系统一启动都会进入到强制模式，但是也提供了可以修改当前安全策略模式的命令。

setenforce 1 设置为强制模式；

setenforce 0 设置为允许模式。

系统三个管理员中只有安全管理员可以使用 setenforce 1 进入到强制模式，

也只有安全管理员可以使用 setenforce 0 进入到允许模式。

（3）禁用安全策略

在系统出现故障时我们也可以先禁用我们的安全策略进入到维护模式进行修复。进入方法如下：

a) 系统启动时进入 grub 菜单。

b) 输入 grub 密码修改内核启动参数，将参数 selinux=1 enforcing=1 修改为 selinux=1 enforcing=0 。

（4）启动系统。

这样系统就直接进入到维护模式，由 root 用户对系统进行修复。

8.2.3.3. 策略配置命令

为了方便用户对安全策略进行配置，我们也提供了以下命令对当前安全策略进行配置：

8.2.3.3.1. 查看标记命令

系统中一些常用命令的-Z 选项都提供了查看文件或进程安全上下文的功能，如：ls -Z id -Z 和 ps -Z，这些命令加了-Z 选项后就会显示该文件或进程的安全上下文，如 ls -Z 一个文件，输出以下内容：

```
-rw-r--r--. root root sysadm_u:object_r:admin_home_t:s0
```

其中 sysadm_u 就表示该文件属于 sysadm_u 安全策略用户，object_r 表示该文件属于客体类型的角色，admin_home_t 表示该文件的类型，我们的策略就是根据这些标记来制定规则进行访问控制的。

8.2.3.3.2. 修改标记命令

安全管理员可以通过一些命令对系统中所有文件的安全上下文进行操作，如：chcon、setfiles 和 restorecon 命令。

/bin/ls 命令为 bin_t 类型，安全管理员可以输入 chcon -t sbin_t /bin/ls 将 bin_t 修改为 sbin_t 类型。

chcon 命令的-u 参数可以修改该文件安全上下文中的安全策略用户，-r 参数可以修改它的角色，-l 参数可以修改它的安全级别，其他具体使用可以查看它的帮助手册。

setfiles 命令可以根据我们的安全策略的标记文件对系统中所有文件进行标记。如：

```
setfiles /etc/selinux/targeted/contexts/files/file_contexts /
```

restorecon 命令可以恢复该文件的默认安全上下文，如：restorecon -r filename。

在对文件的安全上下文标记进行操作时首先都会检查这个标记的正确性，然后再去进行操作，具体使用可以查看用户手册。

8.2.3.3.3. 用户、角色和域的关系

登录程序如（login，sshd）负责映射 Linux 用户到安全策略用户，在登录时，如果安全策略用户标识符恰好和一个 Linux 用户标识符完全相同，匹配的安全策略用户标识符成为初始 shell 进程安全上下文的用户标识符。安全管理员也可使用 semanage 命令完成映射，如：semanage login -a -s charlie_u charlie，表示 Linux 用户 charlie 作为安全策略用户 charlie_u 登录系统。

使用 semanage login -l 命令可以查看，如下图：

```
#semanage login -l

登录名          SELinux 用户        MLS/MCS 范围      服务
__default__        user_u          s0-s0:c0.c1023       *
auditadm          auditadm_u       s0-s0:c0.c1023       *
root               root            s0-s0:c0.c1023       *
secadm            secadm_u         s0-s0:c0.c1023       *
```

一个用户可以对应多个角色，但在同一时刻只能作为其中一个角色，可以通过策略管理工具 semanage 输入 semanage user -a -R mgr_r -R charlie_r -P user charlie_u 来创建一个可以作为角色 mgr_r 和 charlie_r 的安全策略用户 charlie_u。

可通过 semanage user -l 进行查看，如下图：

```
#semanage user -l
                标记中          MLS/        MLS/
SELinux 用户     前缀     MCS 级别     MCS 范围     SELinux 角色
auditadm_u      audadm      s0        s0-s0:c0.c1023    auditadm_r
guest_u         user        s0        s0                guest_r
root            sysadm      s0        s0-s0:c0.c1023    sysadm_r
secadm_u        secadm      s0        s0-s0:c0.c1023    secadm_r
```

角色仅仅是一套域类型的集合，方便与用户建立联系，具体的访问控制都是通过进程的类型和客体的类型根据制定的规则来进行控制。

8.2.3.3.4. 端口配置

系统中所有端口同样也被标记了类型，安全管理员可以通过：

```
semanage port -a -t vnc_port_t -p udp 222
```

来添加一个标记了类型的端口。

可通过 semanage port -l 进行查看，如下图：

```
#semanage port -l
SELinux 端口类型                    协议        端口号
afs3_callback_port_t            tcp      7001
afs3_callback_port_t            udp      7001
afs_bos_port_t                  udp      7007
afs_fs_port_t                   tcp      2040
afs_fs_port_t                   udp      7000, 7005
afs_ka_port_t                   udp      7004
```

8.2.4. 三权分立机制

三权分立系统在满足安全系统标准要求的同时，尽量减少对系统的改动，大大的提高了系统的可用性和引用性。

主要设计思想是，将传统意义上超级管理员 root 的权能进行划分，分别为 root(uid=0)、secadm(uid=600)和 auditadm(uid=700)分别作为超级用户的别名存在，它们分别对应 selinux 三权分立角色中的一个角色，从而保持传统系统用户身份鉴别系统结构的同时，达到三权分立的目的。

三权分立系统是对 linux 现有用户身份认证机制的扩展，增加部分功能主要是为解决传统认证机制无法支持三权分立的用户身份的鉴别。三权系统使用传统 linux 用户身份鉴别机制的认证系统之外可以额外采用双因子验证方式进行身份

鉴别，这套系统是专门定制开发，且符合公安部安全操作系统相关标准。

8.2.4.1. 功能

三权系统功能包括两部分：

1、实现对从一个 UID 为 0 的超级用户衍生出的三个管理员用户的身份认证功能，这个认证系统要支持包括登录系统和用户身份切换等所有可能涉及身份权限切换的情况，同时本三权系统对其他非管理员用户是透明的。

2、实现在每次切换管理员身份时，可以按照不同管理员指派不同的 SELinux 角色给用户。

上面两个功能实现的前提是，本三权系统不能影响到系统中传统身份鉴别系统对非管理员用户的所有验证操作，即新三权系统只对三个管理员用户有作用，对任何其他用户是透明的。

8.2.4.2. 示例

只有系统管理员具有“分区管理功能”。该功能使系统中只有系统管理员可对系统磁盘进行分区管理。当三权系统的安全状态为 strict 模式时。系统管理员可在图形界面下，点击应用程序->工具->磁盘。系统管理员可以成功打开分区编辑器。安全管理员、审计管理员的开始菜单中没有分区编辑器的图标显示。

8.2.4.3. 三权用户命令集

表 8-2 三权用户命令集

用户	命令	备注
系统管理	reboot	系统重启命令。
	init	管理系统运行模式。

员	ifconfig	网络接口管理。
	rpm	软件包管理（但不能删除我们的策略包）。
	useradd	添加用户。如：useradd abc
	passwd	修改普通用户密码。如：passwd abc
	userdel	删除用户。如：userdel -r abc
	insmod	插入模块。
	rmmod	删除模块。
	iptables	防火墙相关。
	system-config-date	配置系统时间（配置网络时间用到 ntpd 服务需要在允许模式下进行）。
	system-config-keyboard	键盘配置。
	system-config-language	语言配置。
	system-config-network system-config-network-cmd system-config-network-tui	网络配置命令。
	system-config-printer system-config-printer-applet	打印机配置。
	setup	整体配置（其中有关服务配置需要切换到允许模式下进行）。
	mount	分区挂载。

	baobab	磁盘分析工具。
安全管理员	setenforce	开启和关闭 selinux 的强制模式 如： setenforce 1 开启强制模式（其他管理员都可执行）； setenforce 0 关闭强制模式（只有安全管理员才能执行）。
	checkpolicy	将可读策略文件编译生成二进制策略文件，该文件和所在文件夹必须为 policy_src_t 类型。 如：checkpolicy -M policy.conf -o policy.24
	load_policy	如：load_policy -bq(服务器)更换内核中的安全策略，保持使用当前的 Boolean 值。
	chcon	修改文件和文件夹的安全上下文。如：chcon -t bin_t filename
	chcat	修改文件和文件夹的敏感级。如：chcat s0:c0.c255 filename
	fixfiles	检查修复文件安全上下文。如：fixfiles -F restore filename
	setfiles	初始化文件标记，检查标记正确性。 如：setfiles

		/etc/selinux/targeted/contexts/files/file_contexts /
	restorecon	恢复默认文件安全上下文。 如：restorecon -r filename
	semanage	SELinux 管理工具命令 如：semanage user -l semanage login -l semanage user -m -R secadm_r secadm_u
	semodule	管理 SELinux 策略模块。 如：semodule -l semodule -i modulename
	setsebool	修改 SELinux 中的 bool 值来修改策略。 如：setsebool xdm_sysadm_login = 1
审计管理员	audit2allow	显示访问被拒绝后应添加的允许规则。如： audit2allow -a audit2allow -i /var/log/dmesg
	audit2why	根据访问被拒绝的审计信息显示被拒绝的原因。 如：audit2why < /var/log/audit/audit.log
	aureport	根据审计信息文件统计登录情况，avc，pid，file，event，config 等众多信息。

		如：aureport -au aureport -a aureport -p
	ausearch	搜索审计文件中的信息。 如：ausearch -gi 0
	auditd	启动审计服务。 如：auditd -s enable
	auditctl	显示服务状态，审计规则相关。 如：auditctl -s
	audispd	审计调度器。 如：audispd

8.2.4.4. 启用与关闭

采用如下方式开启三权分立安全功能：

1）执行下述命令开启三权分立安全功能：

```
security-switch --set strict
```

2）重启系统，使得修改生效

3）通过 security-switch --get 命令查看当前修改后的增强安全状态中“三权分立”为启用，命令如下：

```
security-switch --get
```

显示当前安全级别信息：

```
当前安全级别:strict
-------------------------------------------------------------
  安全模块        |       当前状态        |  默认状态
-------------------------------------------------------------
SELinux              启用(Enforcing)     启用(Enforcing)
三权分立             启用                启用
执行控制             启用(Normal)        启用(Normal)
```

采用切换其他安全模式可以关闭三权分立安全功能：

```
security-switch --set default
```

8.2.5. 核外安全功能及配置

8.2.5.1. 用户 UID 唯一性

操作系统中用户名和用户标识（下称 UID）是存在对应关系，一般接口调用中也是通过 UID 判断对应用户，通常情况下 UID 在用户创建时会自动分配，也可以在创建用户时手动设置未被使用的 UID，该 UID 可以是已删除用户使用过的。而本功能提供对用户 UID 的唯一性检查，确保在整个操作系统生命周期内用户 UID 数字仅且只能使用一次。

若系统已经存在 kylin_1 用户,且该用户下的 UID（可通过命令:id kylin_1 查看 kylin_1 用户的 uid）为 kylin_1-UID，进行如下操作：

```
1.root 删除用户 kylin_1
  #userdel kylin_1
2.root 创建 kylin_2 用户且指定 uid 为 kylin_1-UID
```

```
#adduser kylin_2 –u kylin_1-UID
```

若没有用户 UID 唯一性检查，则添加 kylin_2 用户操作可以成功；该功能使用后步骤 2 中的操作将失败，提示:“adduser：无效的用户 ID”异常信息。

8.2.5.2. 安全切换工具

security-switch 是系统下提供的一个用于安全配置的工具，通过该工具可进行如下 3 种安全模式的切换：

- default 模式：启用系统 kysec 安全机制；
- strict 模式：启用 kysec、selinux、三权分立，且 selinux 为 ukmcs 策略；
- custom 模式:用户自定义安全模式。

8.2.5.2.1. default 模式

设置 default 模式

```
1.kylin 登录系统，执行：
  $sudo security-switch --set default
2.重启系统
```

模式状态：

```
----------------------------------------------------------------
  安全模块        |       当前状态        |  默认状态
----------------------------------------------------------------
 执行控制            用(Normal)        启用(Normal)
```

8.2.5.2.2. strict 模式

设置 strict 模式

1.kylin 登录系统，执行：

#sudo security-switch --set strict

（然后分别设置 root,secadm,auditadm 三个管理员用户的密码）

2.重启系统

模式状态

```
----------------------------------------------------------------
    安全模块        |       当前状态        |  默认状态
----------------------------------------------------------------
  SELinux               启用(Enforcing)     启用(Enforcing)
  三权分立              启用                启用
  执行控制              启用(Normal)        启用(Normal)
```

8.2.5.2.3. 自定义模式

设置自定义模式

1.kylin 登录系统，设置自定义安全机制，仅启用 selinux 安全机制：

#sudo security-switch --set custom –list selinux

2.重启系统；

模式状态

```
----------------------------------------------------------------
安全模块 | 当前状态 | 默认状态
----------------------------------------------------------------
```

```
SELinux 启用(Enforcing) 启用(Enforcing)
```

8.2.5.3. 审计系统权限管理

基于基础安全审计系统功能，麒麟操作系统根据三权分立要求，实现管理员分权机制，修订只允许审计管理员具有审计服务管理权限和审计规则修改权限。根据其系统安全机制要求，增加审计服务容错机制，当审计服务出错时，审计服务将自动重启，确保审计服务正常运行；移除 SU 到审计管理员的权限，限制系统管理员通过 SU 命令执行 auditctl 等审计管理工具。同时，添加了不同等级的审计规则、修复在启用或不启用三权分立时对用户的判断等功能。

审计系统权限需要在安全状态为 strict 模式时相应功能才会开启，另外要确保系统的审计服务处于开启状态（ systemctl statsu auditd ）。

8.2.5.3.1. 登录审计

审计用户登录时，无论成功与否，均会在审计日志中有信息输出。审计日志包括事件类型、时间、用户、事件成功与否、身份鉴别请求的源等，如：

```
  Authentication Report
  ============================================
=======
  #date time acct host term exe success event
  ============================================
========
  2020 年 02 月 09 日  13:21:21 test localhost localdomain
/dev/tty1 /usr/libexec/gdm-session-worker yes 309
```

8.2.5.3.2. 行为审计

添加用户审计规则后，用户的行为会被系统审计。

示例（如对 uid 为 600 的用户添加行为审计规则）：

```
1.审计管理员添加审计规则，对 uid=600 的用户行为进行审计：
#auditctl -a exit,always -S all -F uid=600
2.安全管理员执行操作：
#cat /etc/uid_list
3.审计管理员以 uid 进行审计信息查看
#ausearch -ui 600 | grep uid_list
```

此时可在在审计日志中查看到针对步骤 2 的审计信息。

8.2.5.3.3. 账号管理审计

所有用户账号的增加、删除（useradd、passwd、userdel）等行为都会被系统审计。

示例如下，在审计用户下通过$aureport -m 可以查看到用户账户操作的审计日志。

```
1.root 创建一个用户：
#useradd testuser
2.root 修改用户密码：
#passwd testuser
3.root 删除用户
#userdel testuser
```

8.2.5.3.4. 文件、目录监视审计

添加文件、目录监视规则后，任何对文件、目录的操作都会被系统审计。

示例如下：

存在 kylin 用户创建的/tmp/file1 文件及/tmp/test.dir 目录：

1.审计管理员登录系统，添加审计规则：

$auditctl -w /tmp/file1 -k file

$auditctl –w /tmp/test.dir –k test_dir

2.kylin 用户 cat 查看/tmp/file1 文件

3.kylin 用户使用 rm 删除/tmp/file1 文件

4.kylin 用户对/tmp/test.dir 进行读操作

5.kylin 用户对/tmp/test.dir 进行写操作

6.审计管理员登录系统，删除步骤 1 中审计规则：

$auditctl -W /tmp/file1 -k file

$auditctl –W /tmp/test.dir –k test_dir

步骤 1 添加规则后，步骤 2、3、4、5 等对于文件及目录的操作均会在审计日志中体现；步骤 6 删除规则后不再记录相关操作审计日志。

8.2.5.3.5. 规则配置

添加排除规则后可以排除指定的审计信息。

8.2.5.3.6. 审计日志保护

只有审计管理员才可以查看审计日志。

8.2.5.3.7. 审计日志转储

当审计日志达到设置的最大值时，会覆盖所存储的最早的审计记录。该部分需要审计管理员对配置文件/etc/audit/auditd.conf 进行手动编辑。

示例：

1. 审计管理员编辑/etc/audit/auditd.conf 文件，使

max_log_file=1，num_logs = 3 即日志文件最大为 1MB，备份文件数为 3

2.审计管理员重启服务

#systemctl restart auditd

3.审计管理员添加审计规则，

$auditctl -a exit,always -S all -Fsys

4.安全管理员随便执行任何操作，以便产生大量日志

$find /

5.审计管理员查看审计日志文件大小和数目

$ls -lh /var/log/audit

经步骤 5 查看，系统审计日志有：audit.log、audit.log.1、audit.log.2，其中 audit.log.1、audit.log.2 文件大小都为 1M 左右。

8.2.5.3.8. 审计日志超阈值警报

当审计跟踪的磁盘空间已到达极限时，有审计报警日志提醒。对于磁盘空间极限值的设定，需要审计用户手动调整/etc/audit/auditd.conf。

示例，假设系统日志分区总大小 50G，将 30G 设置为剩余空间阈值，即当用系统日志占用 20G 以上时会有审计日志产生：

1. 审计管理员编辑 /etc/audit/auditd.conf 文件，使 space_left=30000，即日志磁盘空间阈值为 30G；space_left_action=SYSLOG，即向日志中写入一条报警日志；（具体磁盘空间阀值根据测试系统实际日志磁盘空间确定）

2.审计管理员重启服务

```
#systemctl restart auditd
```

步骤 2 后，如果日志空间剩余容量小于 30G，则审计日志中将产生审计日志信息。

8.2.5.4. 支持安全机制审计日志类别

目前审计服务会根据系统中不同的安全机制功能写入不同类型的审计日志类型。

如 kysec 类的审计日志类型为：

```
type=KYSEC_STATUS msg=audit(1502938410.303:182384): status=2 old_status=4 loginuid=0 session=4294967295 id=1502938410.303:182384
 type=KYSEC_AVC msg=audit(1502886638.943:132300): denied { execute } for pid=15325 comm='bash' name='/secadm/ls' ssid=0x01 osid=0x00 loginuid=0 session=4294967295 kysec_status=4 id=1502886638.943:132300
```

8.3. 麒麟安全管理工具-安全中心

安全中心是一款基于麒麟安全框架 KYSEC 的管理工具,提供系统安全加固、账户保护配置、联网控制、应用执行控制和应用防护等功能，保障系统运行环境的安全和稳定。

安全中心集安全加固、账户保护、网络保护和应用保护等功能于一体，全面保障系统运行环境的安全。

安全加固：提供安全服务、内核参数、安全网络、系统命令、系统审计、系统设置、潜在危险、文件权限、风险账户、磁盘检查、密码强度、账户锁定、系

统安全、系统维护、资源分配等多维度的扫描与一键加固，及时发现并处理系统安全隐患。

账户保护：提供系统账户密码强度检查和账户锁定机制，实现对系统账户的统一管控，提升系统账户安全防御能力，有效防止密码被暴力破解。

网络保护：提供应用联网控制功能，实时防护未知应用网络行为，阻断主动外联及其它异常网络活动，提高网络访问安全性。

应用控制：提供应用程序执行控制功能，阻止未知软件、应用程序的恶意执行，避免木马病毒攻击，保障系统运行环境的安全性、可靠性。

应用防护：提供进程防杀死、内核模块防卸载和文件防篡改功能，保护系统关键文件完整性，阻止系统关键应用服务异常中断。

详细内容请参考《银河麒麟安全中心用户手册》。

8.4. 麒麟文件保护箱

麒麟文件保护箱是基于内核级数据隔离机制的保护工具，提供用户间数据隔离和加密保护功能，支持国密算法，实现一箱一密、一文一密的细粒度控制，保障用户数据安全。

具有如下特点：

多重防护：支持用户间数据隔离以及细粒度的权限控制，保障数据安全。

安全加密：支持一箱一密、一文一密的透明加密机制，且对密钥进行安全管理，能够满足政企和金融级客户的核心安全诉求。

丰富算法：支持标准国际算法、国密算法和硬件级加密算法，能够满

足不同安全等级的加密应用场景。

高兼容性：支持保护箱版本兼容机制，用户升级适配无感知，保证用户数据安全存储、永不丢失。

简单易用：支持内置文件管理器，实现统一管理，操作简单、易于上手。

详细内容请参考《银河麒麟文件保护箱用户手册》。

第九章 FAQ

9.1. 版本查询方法

字符终端输入“nkvers”命令即可查询，输出信息即为版本信息，如下所示：

```
#nkvers
###############Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Halberd)

Kernel:
4.19.90-{内核小版本号}..aarch64

Build:
Kylin Linux Advanced Server
release V10 (SP3 2403) /(Halberd)-aarch64-Buildxx/YYYYMMDD
##############################################
```

9.2. 字体安装方法

在安装字体之前首先需要查询系统中已经安装的字体，可以使用 fc-list 命令进行查询，如果系统中没有该命令需要先安装软件包。

```
yum install -y fontconfig mkfontscale
```

执行 fc-list 命令查询已安装的字体，如果发现有缺失相关字体文件，可以将已拿到的字体文件（.ttf、otf 等）放入/usr/share/fonts/目录下，然后使用如下命令建立字体索引信息进行字体安装。

```
cd /usr/share/fonts/
mkfontscale
mkfontdir
fc-cache
```

字体安装完毕之后使用 fc-list 查看安装字体信息，查看字体是否安装成功。

9.3. 详细包信息查询

使用 rpm -pqi 包名，可以查询包详细信息。

```
[root@localhost 桌面]#rpm -pqi tigervnc-1.10.1-3.p02.ky10.x86_64.rpm
Name        : tigervnc
Version     : 1.10.1
Release     : 3.p02.ky10
Architecture: x86_64
Install Date: (not installed)
Group       : Unspecified
Size        : 882530
License     : GPLv2+
Signature   : RSA/SHA1, 2020年12月06日 星期日 07时23分33秒, Key ID 41f8aebe7a486d9f
Source RPM  : tigervnc-1.10.1-3.p02.ky10.src.rpm
Build Date  : 2020年11月13日 星期五 15时40分20秒
Build Host  : localhost.localdomain
```

```
Packager    : Kylin Linux
Vendor      : KylinSoft
URL         : http://www.tigervnc.com
Summary     : A TigerVNC remote display system
Description :
This package provides client for Virtual Network Computing (VNC), with which
you can access any other desktops running a VNC server.
```

9.4. 检查包是否被篡改

使用 rpm -Kv+包名可以查询包是否被篡改，如下都为“OK”

```
[root@localhost 桌面]#rpm -Kv tigervnc-1.10.1-3.p02.ky10.x86_64.rpm
tigervnc-1.10.1-3.p02.ky10.x86_64.rpm:
    头 V3 RSA/SHA1 Signature, 密钥 ID 7a486d9f: OK
    头 SHA256 digest: OK
    头 SHA1 digest: OK
    Payload SHA256 digest: OK
    V3 RSA/SHA1 Signature, 密钥 ID 7a486d9f: OK
    MD5 digest: OK
```