项目 3-拓展资料 Linux NetworkManager 服务介绍

一、Linux 网络设备，是 linux 为网络服务提供的硬件系统。

Linux 设备分为三类：字符设备（如键盘、鼠标等）、块设备（如硬盘、光驱、软驱等）和网络设备，如以太网卡。为了屏蔽网络环境中物理网络设备的多样性，LINUX 对所有的物理设备进行抽象并定义了一个统一的概念，称之为接口（Interface）,接口实际是运行于 Linux 内核一种设备服务程序，所有的硬件厂商按照同一规格编写，从而提供了一个对所有类型的硬件一致化的操作集合来处理基本数据的发送和接收。一个网络接口可以被看作是一个发送和接收数据包（packets）的实体。内核在启动时，通过网络驱动程序，检索系统的固件信息，将网络设备在 Linux 系统的中登记。这样网络设备在做数据包发送和接收时，可以通过接口实现 linux 利用网络设备于外界的交互。接口可以在内核初始化时进行，也可以通过 insmod 命令来加载。

根据标准,每一个具体的网络接口都应该有一个名字，以在系统中能唯一标识一个网络接口。常见的网络接口名如下:

ethN 以太网接口，包括 10Mbps 和 100Mbps；

enoN RHEL7 之后的以太网接口

trN 令牌环接口；

slN SLIP 网络接口；

pppN PPP 网络接口，包括同步和异步；

plipN PLIP 网络接口，其中 N 与打印端口号相同；

dummyN 空设备；

lo 回送网络接口。

如何管理 Linux 网络设备？

Linux 存在很多的网络管理命令，通常我们使用的是 network 服务，NetworkManager 服务， ipconfig 命令， ip 命令 和 netstat 命令，ss 命令等。从 RHEL7 开始， ipconfig，netstat，network 服务已经不再作为系统默认配置，所以下面我们就讲下 NetworkManager 服务

二、NetworkManager 简介：

随着计算机网络技术的进步，网络环境越来越复杂。特别是无线网络技术的发展，其中一个主要的问题是，如何保证在网络环境变化的情况下，保持访问的可持续性。开发者们收集了新时代的网络需求，并开发出了 networkManager。

NetworkManager 的出现主要为了解决下面的问题：软件可以自动检测尽量多的信息，在复杂网络环境下可以平滑的切换，可以即使反馈网络状态，可以在桌面环境下工作。

NetworkManager 由一个管理系统网络连接、并且将其状态通过 D-BUS 进行报告的后台服务，以及一个允许用户管理网络连接的客户端程序组成。

NetworkManager 网络配置顺序

1） 如果有线网络连接可用，NetworkManager 会自动配置有线网络连接。

2） 如果有线网络连接不可以用，NetworkManager 会搜寻所有可检测到的无线网络，如果有保存的授权信息， NetworkManager 会自动配置无线网络连接

3） 如果用户自定义网络信息，用户定义具有最高优先权。

NetworkManger 系统架构

NetworkManger 由四个独立的模块构成：

- NetworkManager 服务 — 一个核心服务，用户处理网络连接和策略

- DHCPCD — NetworkManager 使用的 DHCP 客户端服务 NetworkManagerInfo —

- 收集用户桌面或者客户端定义信息，并反馈网络状态给用户 NetworkManagerNotification —

- Panel 的桌面通知标识符。

这四部分合作，完成 NetworkManager 网络的管理。 他们的结构图如下：

![networkmanager-diagram-en-r1.0.png][1]

NetworkManager 和 D-BUS

D-Bus 是一种被 linux 系统广泛集成的一个内部消息(IPC)和远程进程调用(RPC)组件， 同一系统间，进程可以通过 D-Bus 通过并发交互。D-Bus 是 freedesktop.org 项目的一部分，目的是标准化 linux 桌面（如：KDE，GNOME）的系统消息服务。

NetworkManager 使用 D-BUS 可以增加系统设计的弹性和安全性。

D-BUS 被用于 NetworkManager 内部进程通讯:

- NetworkManager 服务和 NetworkManagerInfo
- NetworkManagerInfo 和 NetworkManagerNotification
- NetworkManager 服务和 HAL

对外，NetworkManager 使用 D-BUS 广播网络状态的变化，并允许通过 D-BUS 改变这种状态

NetworkManager 和 HA

HAL, 全称 硬件抽象曾，它允许应用程序获取硬件的状态。NetworkManager 启动时，查询 HAL 获得可用硬件状态。硬件状态的改变会被 HAL 记录，并传递给 NetworkManager. HAL 可以提供网卡设备的信息，以便

networkManger 使用。

NetworkManager 命令及简单使用

nmcli 的基本配置选项：

Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }

OPTIONS

-t[erse] terse output

-p[retty] pretty output

-m[ode] tabular|multiline output mode

-f[ields] <field1,field2,...>|all|common specify fields to output

-e[scape] yes|no escape columns separators in values

-n[ocheck] don't check nmcli and NetworkManager versions

-a[sk] ask for missing parameters

-w[ait] <seconds> set timeout waiting for finishing operations

-v[ersion] show program version

-h[elp] print this help

OBJECT

g[eneral] NetworkManager's general status and operations

n[etworking] overall networking control

r[adio] NetworkManager radio switches

c[onnection] NetworkManager's connections

d[evice] devices managed by NetworkManager

a[gent] NetworkManager secret agent or polkit agent

nmcli dev 列出所有网络设备

[root@dhcp-129-213 proc]# nmcli dev

DEVICE TYPE STATE CONNECTION

docker0 bridge connected docker0

virbr0 bridge connected virbr0

eno1 ethernet connected eno1

virbr0-nic tap connected virbr0-nic

vnet0 tap connected vnet0

vnet1 tap connected vnet1

lo loopback unmanaged --

nmclie dev disconnect 断开连接

root@dhcp-129-213 proc]# nmcli dev disconnect eno1

Device 'eno1' successfully disconnected.

nmclie dev connect 连接

[root@dhcp-129-213 proc]# nmcli dev connect eno1

Device 'eno1' successfully activated with 'b3a8ec6f-337b-46e6-a88b-b988569fa271'.

nmcli con show 显示设备信息

[root@dhcp-129-213 proc]# nmcli con show

```
NAME                UUID                                  TYPE
DEVICE
docker0             f195ac51-0ad1-425c-b2a2-8964f5625fbc  bridge
docker0
virbr0-nic          a43be4e2-5ab7-43bc-9de4-5a3240edc3c4  generic
virbr0-nic
virbr0              38f0556c-a225-4ff6-9fa7-c7d91744b440  bridge
virbr0
eno1                b3a8ec6f-337b-46e6-a88b-b988569fa271  802-3-ethernet
eno1
vnet1               205ad0f6-7c8e-4e5b-a9ed-231fb248b872  generic
vnet1
vnet0               84c6df62-8163-4ce6-8a46-b00abea19d8c  generic
vnet0
```