Linux系统进阶

Linux软件管理

涉及到安装、卸载、更新和管理软件包。Linux 提供了多种工具和方法来处理软件的安装和管理，不同的 Linux 发行版有不同的包管理工具，同时也支持从源代码安装或使用跨平台的应用程序管理工具。

包管理器

包管理器是 Linux 系统上最常用的工具之一，能够帮助用户简化软件的安装、卸载、更新和管理过程。常见的包管理工具根据 Linux 发行版的不同而有所不同，通常分为两类：基于 `.deb` 的包管理工具和基于 `.rpm` 的包管理工具。

基于 .deb 包管理工具（APT）

APT（Advanced Package Tool）是 Debian 系列及其衍生版（如 Ubuntu）使用的包管理工具。APT 能够自动处理软件包的依赖关系，确保软件包安装或更新时相关依赖项能被正确处理。

安装软件包： 使用 `apt` 命令可以轻松安装软件包。比如，要安装 `curl` 软件包，可以运行：

```
sudo apt update                 # 更新软件包索引
sudo apt install curl           # 安装 curl 软件包
```

卸载软件包： 使用 `remove` 命令卸载软件包，而 `purge` 命令则会同时删除软件包的配置文件。

```
sudo apt remove <package>       # 卸载软件包，保留配置文件
sudo apt purge <package>        # 卸载软件包并删除配置文件
```

更新软件包： 使用 `upgrade` 命令可以将系统中的所有软件包更新到最新版本。

```
sudo apt upgrade                # 更新所有已安装的软件包
sudo apt dist-upgrade           # 升级系统，解决依赖关系，安装/卸载包
```

查找软件包： `apt search` 命令可以搜索可用的软件包：

```
apt search <package>            # 搜索指定的软件包
```

查看已安装的软件包： 要列出系统上已安装的所有软件包，可以使用：

```
apt list --installed            # 列出已安装的软件包
```

基于 .rpm 包管理工具（YUM/DNF）

Red Hat 系列及其衍生版（如 CentOS 和 Fedora）使用 `YUM`（Yellowdog Updater Modified）作为包管理工具，而 Fedora 系统已经改用 `DNF`（Dandified YUM）作为新的包管理工具。YUM 和 DNF 具有相似的命令结构，能够高效地管理系统中的软件包。

安装软件包： 使用 `yum` 或 `dnf` 命令安装软件包。例如，要安装 `httpd` （Apache 服务器）：

```
sudo yum install httpd      # 使用 YUM 安装
sudo dnf install httpd      # 使用 DNF 安装
```

卸载软件包： 可以使用 `remove` 命令卸载软件包：

```
sudo yum remove <package>     # 卸载软件包
sudo dnf remove <package>     # 卸载软件包
```

更新软件包： 使用 `update` 命令可以升级所有已安装的包：

```
sudo yum update              # 使用 YUM 更新软件包
sudo dnf update              # 使用 DNF 更新软件包
```

查找软件包： 使用 `yum search` 或 `dnf search` 可以查找软件包：

```
yum search <package>          # 使用 YUM 搜索
dnf search <package>          # 使用 DNF 搜索
```

查看已安装的软件包：

```
yum list installed            # 使用 YUM 查看已安装软件包
dnf list installed            # 使用 DNF 查看已安装软件包
```

源码编译安装

有时，软件包在发行版的仓库中可能无法找到，或者用户需要安装特定版本的程序。在这种情况下，用户可以从源代码编译安装软件。 正常情况下，需要经历以下步骤：

下载源码包：通常是 `.tar.gz` 或 `.tar.bz2` 格式。

解压源码包：

```
tar -xzvf <package>.tar.gz    # 解压 tar.gz 文件
```

配置源代码： 进入源码目录，运行 `./configure` 进行必要的配置：

```
cd <package>
./configure
```

编译源代码： 使用 `make` 编译源代码：

```
make
```

安装软件： 使用 `sudo make install` 将软件安装到系统中：

```
sudo make install
```

以安装 `Python-2.7.14.tgz` 为例，首先将文件上传至 Kali：

```
┌──(root?kali)-[~]
└─# ls -l Python-2.7.14.tgz
-rw-r--r-- 1 root root 17176758 Oct 11 13:47 Python-2.7.14.tgz
```

开始安装，先解压 `Python-2.7.14.tgz` ：

```
┌──(root㉿kali)-[~]
└─# tar zxvf Python-2.7.14.tgz
Python-2.7.14/
Python-2.7.14/Include/
Python-2.7.14/Include/tupleobject.h
Python-2.7.14/Include/compile.h
Python-2.7.14/Include/metagrammar.h
Python-2.7.14/Include/bytes_methods.h
Python-2.7.14/Include/pythread.h
Python-2.7.14/Include/pystrtod.h
......

┌──(root㉿kali)-[~]
└─# ls -ld Python-2.7.14
drwxr-xr-x 17 kali kali 4096 Sep 16  2017 Python-2.7.14
```

进入源码文件夹，执行 `./configure` 命令，生成 Makefile 文件；

```
┌──(root㉿kali)-[~/Python-2.7.14]
└─# ./configure
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for python2.7... python2.7
checking for --enable-universalsdk... no
checking for --with-universal-archs... 32-bit
checking MACHDEP... linux2
......

┌──(root㉿kali)-[~/Python-2.7.14]
└─# ls -l Makefile
-rw-r--r-- 1 root root 51705 Oct 11 13:56 Makefile
```

执行 `make` 命令将源码自动编译成二进制文件：

```
┌──(root㉿kali)-[~/Python-2.7.14]
└─# make
gcc -c -fno-strict-aliasing -g -O2 -DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prot
otypes  -I. -IInclude -I./Include   -DPy_BUILD_CORE -o Modules/python.o ./Modu
les/python.c
gcc -c -fno-strict-aliasing -g -O2 -DNDEBUG -g -fwrapv -O3 -Wall -Wstrict-prot
otypes  -I. -IInclude -I./Include   -DPy_BUILD_CORE -o Parser/acceler.o Parse
r/acceler.c
......
```

执行 `make install` 安装命令来将上步编译出来的二进制文件安装到对应的目录中去，默认的安装路径为 `/usr/local/`：

```
┌──(root㉿kali)-[~/Python-2.7.14]
└─# make install
/usr/bin/install -c python /usr/local/bin/python2.7
if test -f libpython2.7.a; then \
  if test -n "" ; then \
    /usr/bin/install -c -m 555  /usr/local/bin; \
  else \
......

┌──(root㉿kali)-[~/Python-2.7.14]
└─# ll /usr/local/bin
total 6704
-rwxr-xr-x 1 root root     101 Oct 11 13:58 2to3
-rwxr-xr-x 1 root root      99 Oct 11 13:58 idle
-rwxr-xr-x 1 root root     364 Aug 21 16:27 mount-shared-folders
-rwxr-xr-x 1 root root      84 Oct 11 13:58 pydoc
lrwxrwxrwx 1 root root       7 Oct 11 14:03 python -> python2
lrwxrwxrwx 1 root root       9 Oct 11 14:03 python2 -> python2.7
-rwxr-xr-x 1 root root 6817824 Oct 11 14:03 python2.7
-rwxr-xr-x 1 root root    1687 Oct 11 14:03 python2.7-config
lrwxrwxrwx 1 root root      16 Oct 11 14:03 python2-config -> python2.7-config
lrwxrwxrwx 1 root root      14 Oct 11 14:03 python-config -> python2-config
-rwxr-xr-x 1 root root     338 Aug 21 16:27 restart-vm-tools
-rwxr-xr-x 1 root root   18547 Oct 11 13:58 smtpd.py
```

Linux进程管理

在 Linux 操作系统中，进程是执行中的程序的实例。每个进程都代表了一个独立的任务或活动，它拥有自己的内存空间、执行代码、数据、文件描述符和其他系统资源。

Linux 中提供了丰富的命令和工具来实现进程管理，包括进程的查看、控制、优先级调整等。

进程的概念

在 Linux 中，每一个运行的程序（包括系统守护进程和用户启动的程序）都是一个进程，每个进程在系统中都有一个唯一的进程 ID（PID）作为标识。Linux 进程可以分为两大类：

前台进程：用户在终端直接启动的进程，通常会占用当前终端。

后台进程：在后台运行的进程，不占用当前终端。例如，守护进程（Daemon）通常在后台运行，为系统提供必要的服务。

> **守护进程（Daemon）**是指在后台运行的特殊进程，主要用于在系统启动时自动加载，并持续在后台运行，提供系统服务、资源管理或任务调度等功能。守护进程通常不与用户直接交互，而是默默地在后台等待并响应系统或其他进程的请求。它们是 Linux 系统稳定运行的关键部分，负责处理各类系统任务，例如日志记录、网络服务、任务调度等。

查看进程

ps：显示当前终端会话的进程信息。常用参数：

- `ps aux`：列出系统所有进程，包括用户、进程状态、CPU 和内存使用等。
- `ps -ef`：以详细格式列出所有进程，显示父进程 ID（PPID）等信息。
- `ps -ejH`：查看进程树。

以 `ps aux` 输出为例：

```
USER       PID  %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.2  0.1  18644  3152 ?        Ss   Mar17   0:03 /sbin/init
user      1234  0.5  0.3  20740  4568 pts/0    S    10:05   0:01 bash
user      5678  0.0  0.1  10536  2232 pts/0    R+   10:06   0:00 ps aux
```

各字段解释为：

USER：进程所属的用户名。

PID：进程的 ID（Process ID）。

%CPU：进程的 CPU 使用率，表示该进程使用的 CPU 时间占总 CPU 时间的百分比。

%MEM：进程使用的内存占总内存的百分比。

VSZ：进程使用的虚拟内存大小，单位为 KB。

RSS：进程使用的常驻内存大小（Resident Set Size），单位为 KB。

TTY：进程关联的终端。

STAT：进程的状态，常见的状态有：

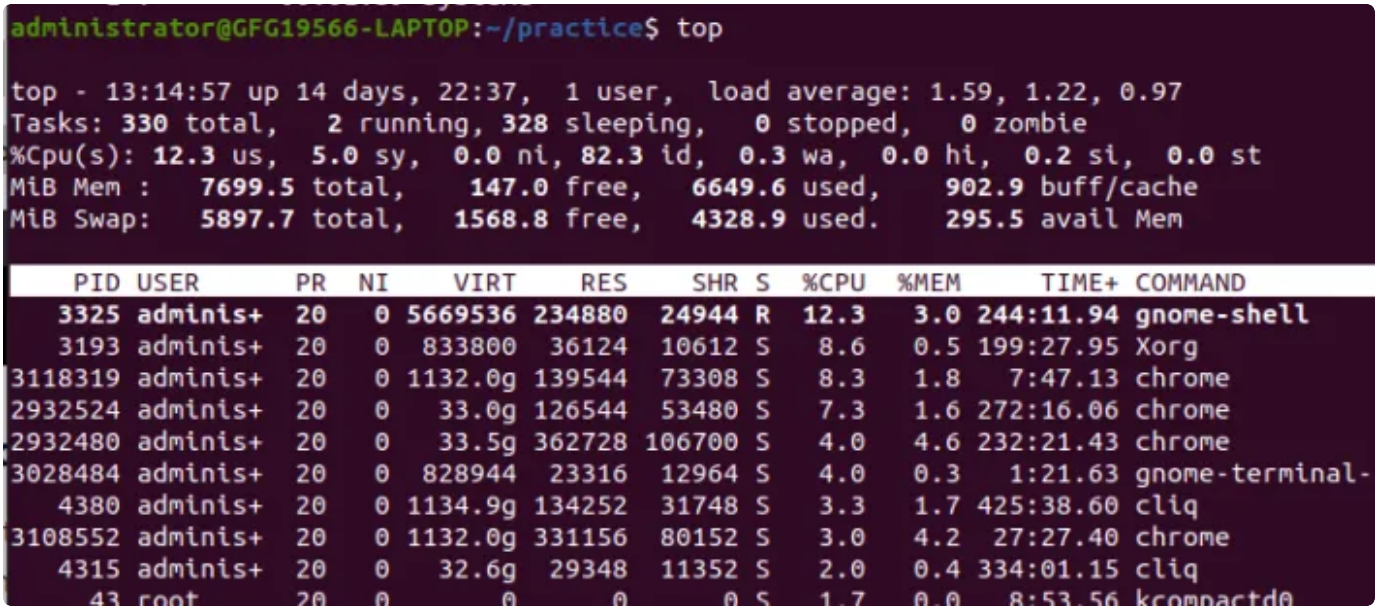
- **S**：休眠状态（Sleeping）
- **R**：运行状态（Running）
- **Z**：僵尸进程（Zombie）
- **T**：停止状态（Stopped）
- **+**：后台进程（后台任务时有此标记）

START：进程的启动时间。

TIME：进程占用的 CPU 时间。

COMMAND：启动进程的命令。

top：实时显示系统的进程信息，包括每个进程的 CPU 和**内存**使用情况。 `top` 命令可以**实时更新**，按 `q` 退出。



pstree： 显示以树状结构排列的进程，方便查看进程之间的父子关系。

lsof

`lsof` 用于列出系统中打开的文件及其关联的进程。由于进程通常会打开文件，使用 `lsof` 可以查看哪些进程占用了系统资源（如文件、套接字等）。

查看某个文件对应的进程：

```
lsof <file_name>
```

查看某个进程打开的文件：

```
lsof -p <PID>
```

控制进程

kill：通过发送信号来控制进程。默认的信号是 `SIGTERM`，用于请求进程正常退出。

- `kill PID`：终止指定 PID 的进程。

- `kill -9 PID`：强制终止进程，发送 `SIGKILL` 信号。

killall：结束指定名称的所有进程。例如， `killall apache2` 会结束所有名为 apache2 的进程。

pkill：通过进程名终止进程，可以使用正则表达式匹配。例如， `pkill -f python` 将结束所有与 Python 脚本相关的进程。

Linux服务管理

在现代 Linux 发行版中，服务管理主要依赖于 `systemctl` 命令， 这是基于 `systemd` 的管理工具 ，广泛用于CentOS 7、Ubuntu 16.04及更高版本。

`service` 命令主要用于较旧的 init 系统（如SysVinit）。

> 在操作系统中，**服务**（Service）是指运行在后台、提供特定功能和响应请求的程序或进程。服务通常在系统启动时启动，独立于用户会话，持续运行，直至手动或系统关闭。

Systemd 介绍

Systemd是现代Linux发行版中广泛使用的初始化系统和服务管理器，主要负责启动、停止和管理系统服务以及系统资源。它最初由Lennart Poettering开发，旨在替代传统的SysVinit系统，并在多个Linux发行版（如Fedora、CentOS、Debian、Ubuntu等）中成为标准。

Systemd的核心功能

并行启动: Systemd可以同时启动多个服务，从而加快系统的启动速度。

依赖关系管理: 它能够自动处理服务之间的依赖关系，确保在启动服务时先启动其依赖的服务。

单元（Unit）管理: Systemd使用“单元”来管理服务、挂载点、设备等。每个单元都有一个配置文件，定义了服务的启动、停止、重启等行为。

关键概念

单元（Unit）：Systemd管理的基本对象，可以是：

- **服务单元（.service）**：管理系统服务的单元。
- **挂载单元（.mount）**：管理文件系统挂载点的单元。

- **设备单元（.device）**：管理设备的单元。
- **目标单元（.target）**：用于组织和管理多个单元的逻辑集合，类似于传统的运行级别。

常用Systemd命令

启动服务:

```
sudo systemctl start service_name
```

停止服务:

```
sudo systemctl stop service_name
```

重启服务:

```
sudo systemctl restart service_name
```

查看服务状态:

```
sudo systemctl status service_name
```

设置服务开机自启:

```
sudo systemctl enable service_name
```

禁用服务开机自启:

```
sudo systemctl disable service_name
```

列出所有服务:

```
sudo systemctl list-units --type=service
```

Sysvinit

SysVinit是一个传统的Linux初始化系统，用于启动和管理系统服务和进程。它通过一系列脚本来管理系统的启动、关机和运行级别，广泛应用于较早的Linux发行版。尽管在许多现代发行版中已被Systemd取代，SysVinit仍然在某些环境中使用，特别是在较老的系统中。

SysVinit的主要命令

启动服务:

```
sudo service service_name start
```

停止服务:

```
sudo service service_name stop
```

重启服务:

```
sudo service service_name restart
```

查看服务状态:

```
sudo service service_name status
```

Linux网络管理

网络接口管理

网络接口管理是网络管理的基础，网络接口就是计算机与其他设备或计算机网络通信的通道。在Linux中，每个物理网络接口都有一个唯一的标识符，如 `eth0` 、 `eth1` ，无线接口一般是 `wlan0` 或 `wlp2s0` 。每个网络接口都可以配置为静态IP或动态获取IP。 可以使用 `ifconfig` 命令查看和配置网络信息， `lo` 是本地回环接口，通常用于计算机与自身通信 。

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.134.208  netmask 255.255.255.0  broadcast 192.168.13
4.255
        inet6 fe80::3354:4d64:9f99:d4fd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:07:4e:f4  txqueuelen 1000  (Ethernet)
        RX packets 1814  bytes 1597828 (1.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1065  bytes 144785 (141.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

```

`ip` 命令是 `ifconfig` 的替代工具，更加现代和强大。它可以进行网络接口的管理、路由的配置、IP地址的管理等。Linux推荐使用 `ip` 命令来取代 `ifconfig` 。

查看接口信息：

Plain Text

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defa
ult qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 00:0c:29:07:4e:f4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.134.208/24 brd 192.168.134.255 scope global dynamic noprefixr
oute eth0
       valid_lft 1546sec preferred_lft 1546sec
    inet6 fe80::3354:4d64:9f99:d4fd/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

网络服务管理

网络服务是指各种网络应用程序，如 Web 服务（HTTP）、DNS 服务、FTP 服务等。Linux系统通过 `ss` 、 `netstat` 等命令来查看和管理网络服务的状态。管理员可以通过这些工具检查端口是否开放，是否有服务在监听，进而对网络进行维护。

ss：这是现代Linux系统中用于查看网络连接的工具，它比 `netstat` 更加高效。使用 `ss` 可以查看当前系统的TCP、UDP连接信息，了解哪些服务在监听哪些端口。

使用 `ss -tuln` 查看所有监听的端口：

```
State      Recv-Q Send-Q   Local Address:Port    Peer Address:Port
LISTEN     0      128          0.0.0.0:22           0.0.0.0:*
LISTEN     0      128          0.0.0.0:80           0.0.0.0:*
ESTAB      0      0            192.168.1.100:22     192.168.1.101:32456
```

具体解释：

State：表示套接字的状态（例如： `LISTEN` 、 `ESTAB` 、 `CLOSE_WAIT` 、 `TIME_WAIT` 等）。

- `LISTEN` ：表示该端口正在监听来自其他设备的连接请求。

- `ESTAB`：表示该连接已建立，正在进行数据传输。
- `CLOSE_WAIT`、`TIME_WAIT` 等：这些是与 TCP 连接关闭相关的状态。

Recv-Q：接收队列中等待处理的数据字节数。表示套接字接受数据的缓冲区的使用情况。如果为 0，表示没有待处理的接收数据。

Send-Q：发送队列中待发送的数据字节数。表示套接字发送数据的缓冲区的使用情况。如果为 0，表示没有待发送的发送数据。

Local Address：本地 IP 地址和端口，表示套接字在本机的地址。

- `0.0.0.0:80` 表示该套接字在所有 IP 地址上监听端口 80。
- `192.168.1.100:22` 表示该套接字在本机 IP 地址 `192.168.1.100` 上监听端口 22。

Peer Address：对端（远程）IP 地址和端口，表示与之建立连接的另一端的地址。如果是监听状态，该列通常显示为 `*`。

netstat：这是一个传统的工具，用来显示网络连接、路由表和接口信息等。 常用的选项组合有 `netstat -tuln` 和 `netstat -an`

防火墙管理

Linux提供了强大的防火墙管理工具，允许管理员通过配置防火墙规则来控制入站和出站流量，保护系统免受未经授权的访问。防火墙配置工具包括 `iptables`、`firewalld` 等。

iptables：`iptables` 是一个基于规则的防火墙工具，它通过定义一系列规则来过滤和控制流入和流出的网络流量。管理员可以使用 `iptables` 来设置访问控制、网络地址转换（NAT）等。

查看现有的规则：

```
1   iptables -L
```

允许TCP端口80：

```
1   iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

firewalld：`firewalld` 是一种基于区域的动态防火墙管理工具，用于较新的Linux系统。`firewalld` 是通过定义不同的区域来管理流量的，常用于CentOS 7及更高版本。

查看防火墙状态：

```
1   firewall-cmd --state
```

添加允许的端口：

```
firewall-cmd --add-port=80/tcp --permanent
firewall-cmd --reload
```

拓展

Kali换源：https://www.perfcode.com/kali/config/update-source