07 系统高级管理

进程与程序

程序是一组指令的有序集合，是包含可执行代码的形态文件。

进程由程序产生，是动态的，是一个运行着的、要占用系统资源的程序。

进程和程序区别：

(1)进程是动态的，而程序是静态的。

(2)进程有一定的生命期，而程序是指令的集合。

(3)一个程序可以启动多个进程。

(4)多个进程可以调用同一个程序。

进程的分类

进程一般分为交互进程、批处理进程和守护进程三类：

- 交互进程：由shell启动的进程，可以在前台或后台运行。
- 批处理进程：这种进程和终端没有联系，是一个进程序列。
- 守护进程：在后台运行，并且没有控制终端的进程。通常随着操作系统的启动而运行，也将其称为服务。例如，apache服务器的守护进程。

```
lei@lei-VirtualBox:~$ ps aux | grep apache2
root      1033  0.0  0.4 371460 18480 ?        Ss   14:46   0:00 /usr/sbin/apache2 -k start
www-data  1071  0.0  0.2 373784 10532 ?        S    14:46   0:00 /usr/sbin/apache2 -k start
www-data  1072  0.0  0.2 373784 10532 ?        S    14:46   0:00 /usr/sbin/apache2 -k start
www-data  1073  0.0  0.2 373784 10532 ?        S    14:46   0:00 /usr/sbin/apache2 -k start
www-data  1074  0.0  0.2 373784 10532 ?        S    14:46   0:00 /usr/sbin/apache2 -k start
www-data  1075  0.0  0.2 373784 10532 ?        S    14:46   0:00 /usr/sbin/apache2 -k start
```

前台与后台

- 前台是指一个程序控制着标准输出和标准输入。
- 后台就是指一个程序不从标准输入接受输入，一般也不将结果输出到标准输出上。
- 一般地，用户键入一个命令，就已经启动了一个前台的进程。
- 对于非常耗时的进程，可以让进程在后台运行。从后台启动进程其实就是在命令的结尾处加上一个“&”号。

LINUX进程

进程的三种基本状态：

运行态（Running）： 进程占有CPU，并在CPU上运行

就绪态（Ready）：一个进程已经具备运行条件，但由于无CPU暂时不能运行的状态（当调度给其CPU时，立即可以运行）

等待态（Blocked）：阻塞态、封锁态、睡眠态指进程因等待某种事件的发生而暂时不能运行的状态（即使CPU空闲，该进程也不可运行）

进程的属性

一个进程主要有以下参数：

- PID（进程号）：用于唯一标识进程；
- PPID（父进程号）：创建某进程的上一个进程的进程号；
- USER：启动进程的用户ID和该用户所属的组的ID；
- STAT：进程状态，状态分为运行、等待、停止、休眠、僵尸等；
- PRIORITY：进程执行的优先级；
- 进程资源占用：进程占用资源大小（比如内存、CPU占用量）。

查看进程状态

ps 命令可以用于确定正在运行的进程，进程的状态，进程占用的资源等。ps命令提供的是进程的一次性查看，结果不是动态连续的。

命令格式：PS [选项]

ps 命令常用的选项有以下几个：

- -l　　按长格式显示；
- -f　　按全格式显示进程（列出进程间父子关系）；
- -e　　显示所有进程；
- a　　显示所有用户的进程；
- u　　显示进程所有者的信息；
- x　　显示没有控制终端的进程及后台进程；
- r　　显示运行中的进程；

例子：

```
#ps
#ps –aux | less
#ps –aux | grep bash
#ps -p $$ 显示当前使用shell
```

PS命令

最常用的组合是aux

```
lei@lei-VirtualBox:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 160112  9432 ?        Ss   Sep30   0:02 /sbin/init spla
root           2  0.0  0.0      0     0 ?        S    Sep30   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Sep30   0:00 [rcu_gp]
```

USER 进程的所有者；

PID 进程的ID；

%CPU 进程占用的CPU百分比；

%MEM 占用内存的百分比；

VSZ 占用虚拟内存的数量；

RSS 驻留内存的数量；

TTY 进程的控制终端（？表示该进程与终端没有关联）；

STAT 进程的运行状态；

START 进程开始时间；

TIME 进程已经执行的时间；

COMMAND 进程对应的程序名称和运行参数；

进程状态

STAT 进程运行状态

- R 正在运行或在队列中等待运行状态;
- S 可中断的休眠状态;
- D 不可中断的休眠状态;
- W 无足够内存页面可分配;
- < 高优先级的进程;
- N 低优先级的进程;
- + 一个前台进程组;
- l 多线程进程 ;
- s 创建会话的进程;
- L 内存页面被锁定。

动态查看进程信息

top命令

- 和ps 相比，top是动态监视系统进程信息的工具，top 输出的结果是连续的 ，
- 格式为：top [选项]

常用选项：

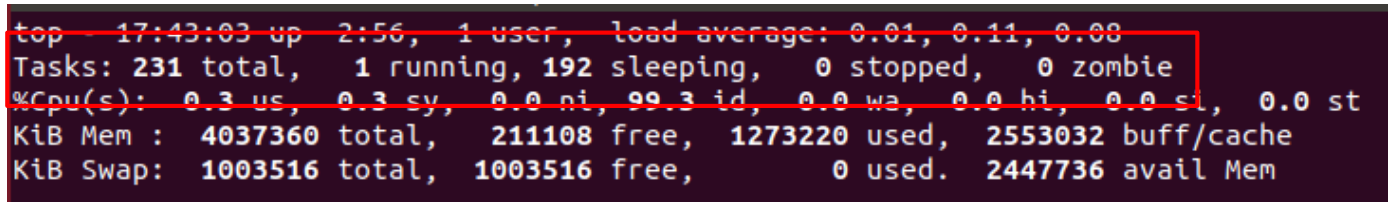
- -d N 显示两次刷新时间的间隔，默认5秒。比如 -d 5，表示两次刷新间隔为5秒；
- -s 表示top在安全模式中运行，不能使用交互命令；
- -c 显示命令行，而不仅仅是命令名；

```
top - 17:36:26 up  2:49,  1 user,  load average: 0.75, 0.33, 0.12
Tasks: 229 total,   1 running, 188 sleeping,   0 stopped,   0 zombie
%Cpu(s):  7.7 us,  1.0 sy,  0.0 ni, 91.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  4037360 total,   368104 free,  1123804 used,  2545452 buff/cache
KiB Swap:  1003516 total,  1003516 free,        0 used.  2598192 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
27503 lei       20   0 3020380 284376 110888 S  6.6  7.0   0:05.34 gnome-shell
27356 lei       20   0  438060  99360  53292 S  0.7  2.5   0:00.97 Xorg
27835 lei       20   0  866896  37012  28136 S  0.7  0.9   0:00.32 gnome-termi+
27455 lei       20   0  126388   2304   1932 S  0.3  0.1   0:00.06 VBoxClient
    1 root      20   0  160272   9820   6952 S  0.0  0.2   0:06.51 systemd
```

```
top - 17:43:03 up  2:56,  1 user,  load average: 0.01, 0.11, 0.08
Tasks: 231 total,   1 running, 192 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  4037360 total,   211108 free,  1273220 used,  2553032 buff/cache
KiB Swap:  1003516 total,  1003516 free,        0 used.  2447736 avail Mem
```

第1行是任务队列信息，其参数如下：

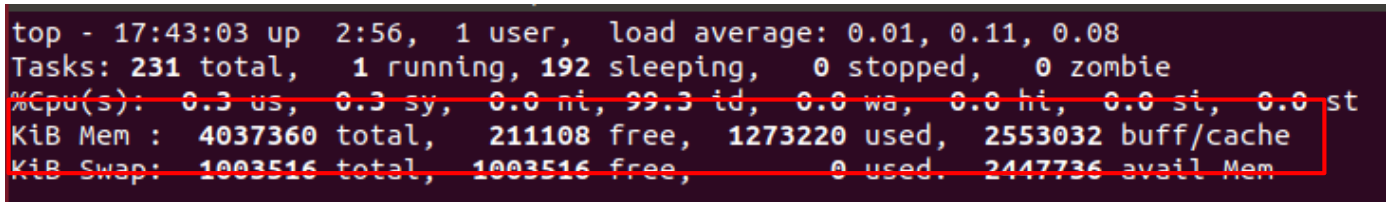
内容	含义
17:43:03	表示当前时间
up 2:56	系统运行时间 格式为时：分
1 users	当前登录用户数
load average: 0.01, 0.11, 0.08	系统负载，即任务队列的平均长度。 三个数值分别为 1分钟、5分钟、15分钟前到现在的平均值。



Tasks：232 total	总进程数
1 running	正在运行的进程数
192 sleeping	睡眠的进程数
0 stopped	停止的进程数
0 zombie	僵尸进程数，如果不是0，需要手工检查僵尸进程

```
top - 17:43:03 up  2:56,  1 user,  load average: 0.01, 0.11, 0.08
Tasks: 231 total,   1 running, 192 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  4037360 total,   211108 free,  1273220 used,  2553032 buff/cache
KiB Swap:  1003516 total,  1003516 free,        0 used.  2447736 avail Mem
```

Cpu(s): 0.3 us	用户空间占用的CPU百分比
0.3 sy	内核空间占用的CPU百分比
0.0 ni	改变过优先级的用户进程占用的CPU百分比
99.3 id	空闲CPU的百分比
0.0 wa	等待输入/输出的进程所占用的百分比
0.0 hi	硬中断请求服务占用的CPU百分比
0.0 si	软中断请求服务占用的CPU百分比
0.0 st	虚拟机里外借的CPU百分比，物理机系统不会出现这个参数



KiB Mem: 4037360 total	物理内存总量
211108 free	空闲的内存总量
1273220 used	已使用的物理内存总量
2552032 buff/cache	作为缓冲的内存数量

```
top - 17:43:03 up  2:56,  1 user,  load average: 0.01, 0.11, 0.08
Tasks: 231 total,   1 running, 192 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  4037360 total,   211108 free,  1273220 used,  2553032 buff/cache
KiB Swap:  1003516 total,  1003516 free,        0 used.  2447736 avail Mem
```

KiB Swap：1003516 total	交换分区的总大小
1003516 free	空闲的大小
0 used	已使用的大小
2447736 avail Mem	代表可用于进程下一次分配的物理内存数量

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
27503	lei	20	0	3024304	299020	111004	S	5.6	7.4	0:14.52	gnome-shell
27356	lei	20	0	439084	99592	53524	S	0.7	2.5	0:02.91	Xorg
27455	lei	20	0	126388	2304	1932	S	0.3	0.1	0:01.85	VBoxClient
27698	lei	20	0	1060044	58156	43824	S	0.3	1.4	0:01.05	nautilus-de+
27854	**lei**	**20**	**0**	**49004**	**3900**	**3172**	**R**	**0.3**	**0.1**	**0:03.79**	**top**
1	root	20	0	160272	9820	6952	S	0.0	0.2	0:06.53	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd

PID 进程号；

USER 进程的所有者；

PR 优先级；

NI 进程的NICE值（负值表示优先级高，正值表示优先级低）；

VIRT 进程使用的虚拟内存总量（单位kb）；

RES 进程使用的未被换出的物理内存大小（单位kb）；

SHR 共享内存大小；

S 进程的运行状态；

%CPU 进程占用的CPU百分比；

%MEM 占用内存的百分比；

TIME+ 进程使用的CPU时间（单位1/100秒）；

COMMAND 进程对应的程序名称和运行参数。

交互式命令键

交互式命令键：

- space 立即更新；
- k 提示输入要杀死的进程ID，目的是用来杀死该进程（默认信号为15）
- i 禁止空闲进程和僵尸进程；
- l 切换到显示负载平均值和正常运行的时间等信息；
- m 切换到内存信息，并以内存占用大小排序；
- n 提示显示的进程数，比如输入3，就在整屏上显示3个进程；

交互式命令键

- r 把renice 应用到一个进程，提示输入PID和renice的值；
- s 改变两次刷新时间间隔，以秒为单位；
- t 切换到显示进程和CPU状态的信息；
- A 按进程的生命进行排序，最新进程显示在最前；
- M 按内存占用大小排序，由大到小；
- N 以进程ID大小排序，由小到大；
- P 按CPU占用情况排序，由大到小。

启动进程

启动进程需要运行程序，有两个主要途径：手动启动和调度启动。

1. 手动启动

- 前台进程启动：程序名 <回车>
- 后台进程启动：程序名 & <回车>
- 注：交互进程在用户注销时自动关闭，为了让注销时程序能够继续运行，可以 nohup 程序名 &。

2. 调度启动（at, cron, anacron等）

事先设定好程序要运行的时间，到了预设时间后，系统自动启动程序。

进程的挂起与恢复

通常将正在执行的一个或多个相关进程称为一个作业（job）。作业控制指的是控制正在运行的进程的行为，可以将进程挂起并在需要时恢复。

- <ctrl>+z 挂起当前的前台作业，把程序放到后台。
- jobs 查看挂起的作业。
- fg和bg
- fg %number
  bg %number
  参数说明：
  %:后面接数字，表示 jobs 的工作代号
  number ： 就是工作代号
- 一般先使用 jobs 来看看后台程序的代号，然后以 bg %number 来使程序在后台中执行，而 fg %number 则是将代号为 number 的程序移动到前台来运行！

```
# showdate.sh
while true
do
   echo $(date)
done
```

结束进程

当一个程序已经死掉，但又不能退出，这时就应该考虑结束进程。组合键<ctrl>+c可以中断一个前台进程。对于后台进行，可以使用kill命令。

1. kill命令是通过向进程发送指定信号来结束进程的。

kill 基本用法：

kill [-s，--信号] 进程号

选项-s指定需要送出的信号，可以是信号名，也可以是对应的数字。默认为TERM信号。

使用ps命令获得进程的进程号，比如获取bash的进程号：

ps –e | grep bash

假设某进程的PID为1234，终止该进程可以执行

kill -9 1234

信号

信号简称	数值	代表意义
SIGHUP	1	端挂起或者控制进程终止
SIGINT	2	键盘的插入指令(同 Ctrl + C)
SIGQUIT	3	键盘的中断指令(同 Ctrl + \)
SIGTERM	15	程序的终止指令
SIGKILL	9	程序的强制终止指令
SIGCONT	18	程序的再启动指令(STOP(19) 后再重新启动)
SIGSTOP	19	程序的停止指令(同 Ctrl + Z)

- 常用的信号代码是-9， 表示强制终止。

按进程名结束进程

2. killall命令

- killall 通过程序的名字，直接杀死所有进程。

  用法：killall 程序名

  例如：

  ps –aux | grep mozi*

  killall mozilla-bin

管理进程优先级

- 在Linux中，进程之间是竞争资源（比如CPU和内存的占用）的关系。这个竞争优劣是通过一个数值来实现的，也就是谦让度。高谦让度表示进程优化级别低。负值或0表示较高优先级，对其它进程不谦让，也就是拥有优先占用系统资源的权利。谦让度的值从-20到19。

- 目前硬件技术发展极速，在大多情况下，不必设置进程的优先级，除非在进程失控而疯狂占用资源的情况下，我们有可能来设置一下优先级。
- nice 可以在创建进程时，为进程指定谦让度的值，进程的优先级的值是父进程SHELL的优先级的值与我们所指定谦让度的相加和。所以我们在用nice设置程序的优先级时，所指定数值是一个增量，并不是优先级的绝对值；

设置进程优先级

- 使用ps -l 来查看显示出来的进程的信息。
- PRI 代表这个程序“可被执行的优先级”越小越早被执行!
- NI 代表这个程序的 nice 值!
- PRI(new) = PRI(old) + nice
- 一般使用者可用 nice 值： 0 ~ 19
- root 管理员可用 nice 值： -20 ~ 19

设置进程优先级

1. nice 用于设置进程的优先级。

语法：nice [-n number] command

参数说明：

-n ： 就是后面那个number即为nice值，默认为10。

范例：

nice -n -5 command

2. renice 用于调整进程的优先级，只有root权限才能使用。

语法：renice [number] PID

范例：

ps -aux | grep showdate

renice -5 27821

top –p 27821

```
lei@lei-VirtualBox:~$ ps -aux | grep showdate
lei       27821  2.1  0.0  19988  3296 pts/0    T<   18:40   0:12 bash showdate.s
h
lei       29143  0.0  0.0  21532  1096 pts/0    S+   18:50   0:00 grep --color=au
to showdate
lei@lei-VirtualBox:~$ renice -5 27821
27821 (process ID) old priority -20, new priority -5
```

```
  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
27821 lei       15  -5   19988   3296   3032 T  0.0  0.1   0:12.48 bash
```

系统启动过程

1. 加电自检(加载BIOS)

- 主板BIOS程序加载；
- 检查所有硬件是否正常工作；
- 搜索处于活动状态并且可以引导的设备；
- 引导设备可以是U盘、光盘、硬盘、网卡，通常从硬盘引导。

BIOS，基本输入输出系统，存储在计算机主板上的EEPROM或闪存中，它是计算机最重要的基本输入输出的程序、开机后自检程序和系统自启动程序，它可以从CMOS中读写系统设置的具体信息。

CMOS是主板上的一块可读写的并行或串行FLASH芯片，是用来保存BIOS的硬件配置和用户对某些参数的设定。

系统启动过程

2. 启动引导加载程序（boot loader）

- 选择引导设备后，就读取该设备的MBR(主引导记录)引导扇区；
- 当MBR加载到内存后，BIOS将控制权交给MBR；
- MBR启动引导加载程序，由其引导操作系统；
- Ubuntu使用GRUB作为默认引导加载程序。

GRUB是大多数Linux发行版本的引导程序，在启动的时候，GRUB会显示一个菜单列表供用户选择。

系统启动过程

3. 装载内核（kernel）

- GRUB载入Linux系统内核并运行；
- 初始化设备驱动程序；
- 以只读方式挂载根文件系统。

系统启动过程

4. 执行init程序实现系统初始化

- 加载init程序。init是第一个运行的进程（进程号是1），是所有进程的发起者和控制者；
- 依靠进程init完成系统初始化，进入特定的运行级别运行相应的程序和服务；
- 提供用户登录界面。

系统启动过程

5. 用户登录

- login处理用户的登录操作；
- 调用shell，运行相关启动文件，初始化各种必要的变量，设置运行环境；
- Bash拥有的启动文件包括：/etc/profile，/etc/bash.bashrc，~/.bashrc和~/.profile。

引导文件

在系统启动过程中，从启动引导加载程序开始，到加载内核之前都有GRUB（Grand Unified Bootloader）负责。

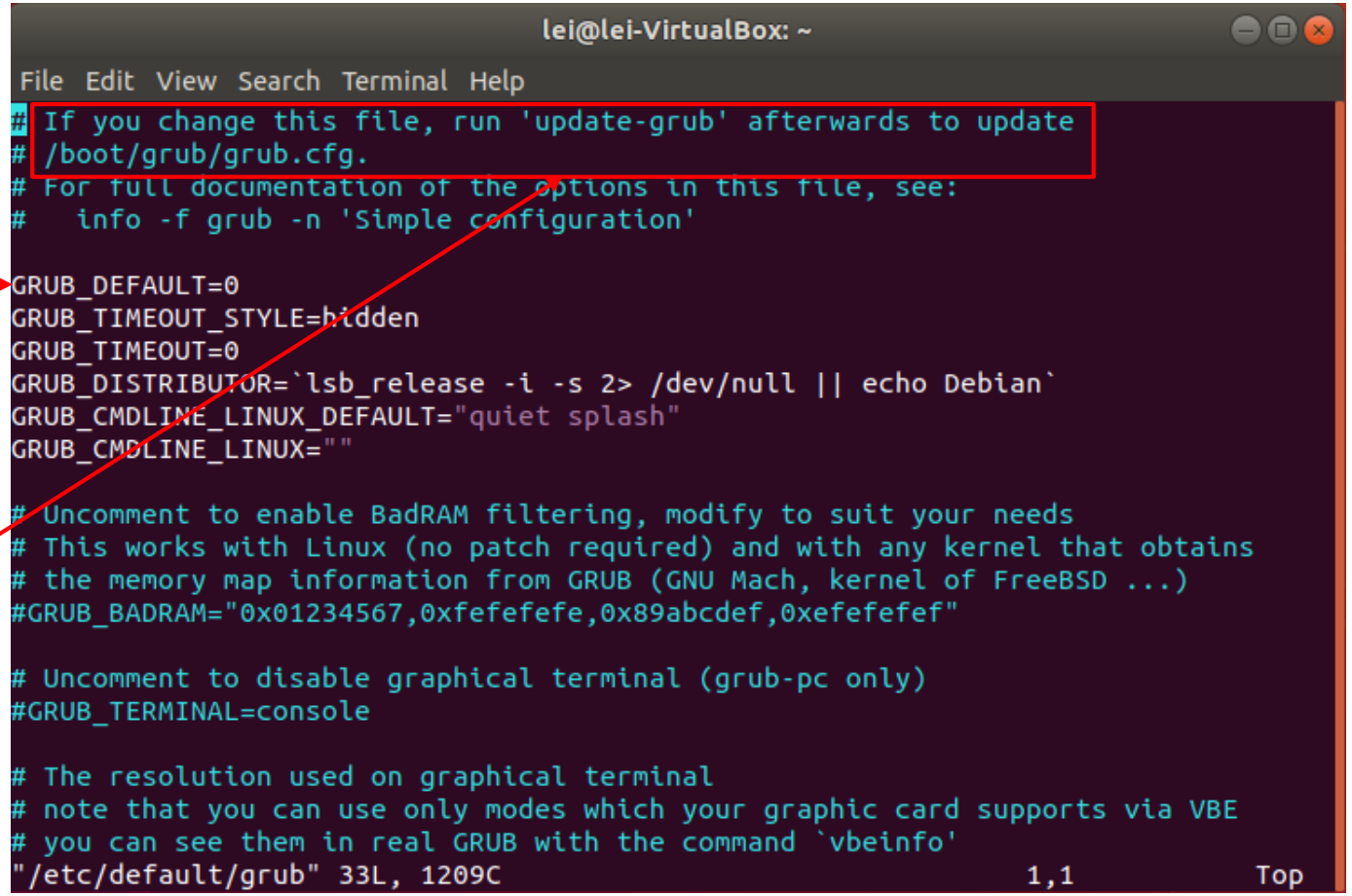
GRUB配置文件/boot/grub/grub.cfg，该文件由grub-mkconfig命令根据/etc/grub.d中的模板和/etc/default/grub中的设置自动生成。通过GRUB引导时会自动读取该配置文件，进而显示引导画面。

可以通过修改/etc/default/grub达到修改GRUB的目的，修改后保存，然后执行sudo update-grub更新。

引导文件

默认启动菜单项

修改后运行update-grub更新



UBUNTU运行级别

运行级别（runlevel）就是操作系统当前正在运行的功能级别。Linux使用运行级别来设置不同环境下所运行的程序和服务。

Ubuntu的默认开机的runlevel是5，可以用runlevel来查看当前的默认运行级别。

级别	Ubuntu	Redhat
0	关机模式	关机
1	单用户模式	单用户模式
2	带显示管理器(GUI)的完整多用户模式	多用户模式
3	带显示管理器(GUI)的完整多用户模式	字符界面的多用户模式
4	带显示管理器(GUI)的完整多用户模式	未定义
5	带显示管理器(GUI)的完整多用户模式	图形界面的多用户模式
6	重启	重启

init进程

init进程完成大量系统初始化工作，运行各种服务。

Linux系统有三种init方式：

- System V initialization(SysVinit) 传统方式。
- Upstart
- systemd

SysVinit是一个基于运行级别的系统，它使用运行级别和对应的链接文件来启动和关闭系统服务。

```
lei@lei-VirtualBox:/etc$ ls rc
rc0.d/ rc1.d/ rc2.d/ rc3.d/ rc4.d/ rc5.d/ rc6.d/ rcS.d/
lei@lei-VirtualBox:/etc$ cd rc2.d/
lei@lei-VirtualBox:/etc/rc2.d$ ls
S01acpid             S01cron           S01irqbalance  S01saned
S01anacron           S01cups           S01kerneloops  S01speech-dispatcher
S01apport            S01cups-browsed   S01mysql       S01spice-vdagent
S01avahi-daemon      S01dbus           S01plymouth    S01unattended-upgrades
S01bluetooth         S01gdm3           S01rsync       S01uuidd
S01console-setup.sh  S01grub-common    S01rsyslog     S01whoopsie
```

init进程

Upstart是基于事件机制的启动系统，它使用事件来启动和关闭系统服务。事件可以由系统内部产生，也可以由用户提供。它不仅能在运行级别改变的时候启动或停止服务，也能在接收到系统发生其它改变的信息时启动或停止服务。

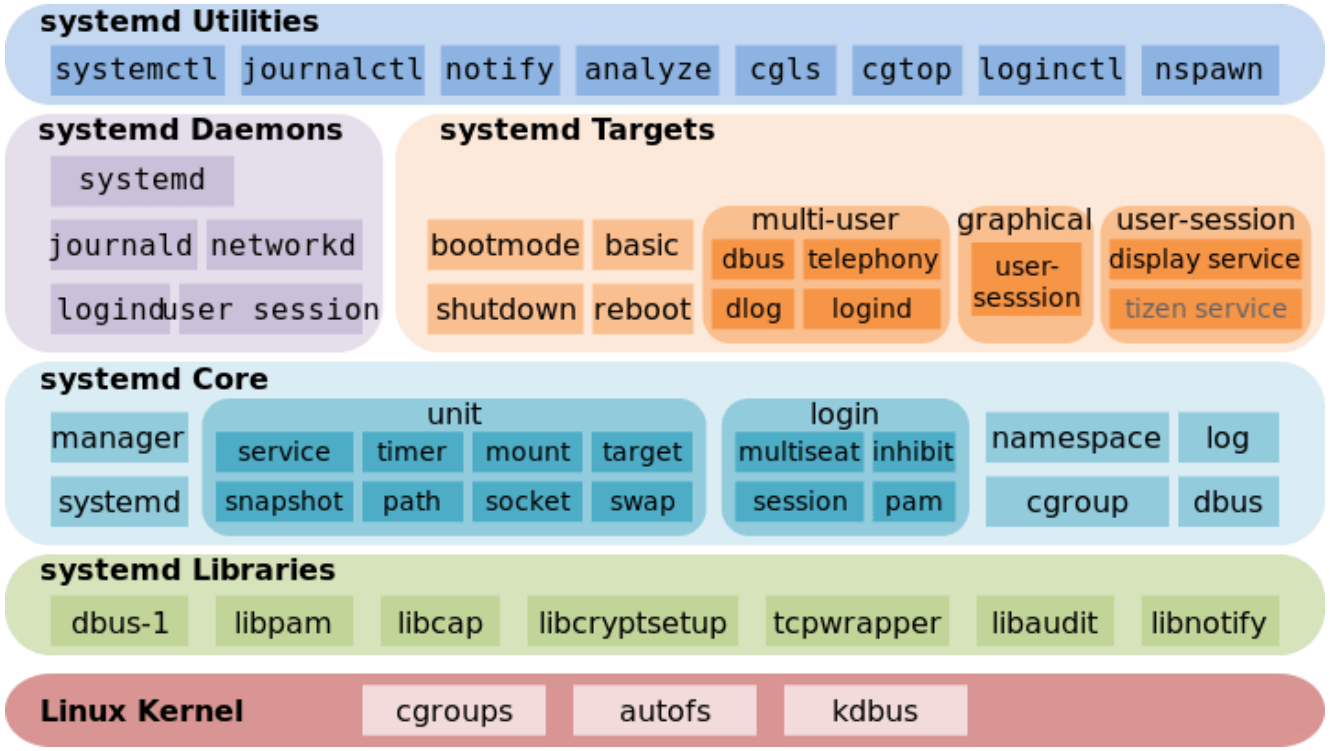
Upstart使用/etc/init目录中的系统服务配置文件来决定系统服务何时启动，何时停止。 /etc/init目录中每个文件定义一个服务或作业，例如mysql.conf

在运行级别2345时启动

```
description     "MySQL 5.7 Server"
author          "Mario Limonciello <superm1@ubuntu.com>"

start on runlevel [2345]
stop on starting rc RUNLEVEL=[016]

respawn
respawn limit 2 5

env HOME=/etc/mysql
umask 007

# The default of 5 seconds is too low for mysql which needs to flush buffers
kill timeout 300
```

init进程

Systemd是为了解决前两种初始化程序启动时间长和启动脚本复杂的缺点而设计的，它的设计目标是为系统的启动和管理提供一套完整的解决方案。它不仅是Linux系统和服务的管理工具，而且还可以作为其它软件的基础平台。



systemd

Systemd是第一个启动的进程，进程ID为1，而在系统关闭过程中，systemd是最后停止的进程。

使用pstree以树形方式显示正在运行的所有进程

```
lei@lei-VirtualBox:~$ pstree -p
systemd(1)─┬─ModemManager(728)─┬─{ModemManager}(803)
           │                   └─{ModemManager}(848)
           ├─NetworkManager(845)─┬─dhclient(963)
           │                     ├─{NetworkManager}(891)
           │                     └─{NetworkManager}(893)
           ├─VBoxClient(1461)───VBoxClient(1462)
           ├─VBoxClient(1630)───VBoxClient(1631)───{VBoxClient}(1666)
           ├─VBoxClient(1640)───VBoxClient(1641)
           ├─VBoxClient(1647)───VBoxClient(1648)───{VBoxClient}(1649)
           ├─VBoxClient(1654)───VBoxClient(1655)─┬─{VBoxClient}(1656)
           │                                     └─{VBoxClient}(1658)
           ├─VBoxService(1486)─┬─{VBoxService}(1487)
           │                   ├─{VBoxService}(1488)
           │                   ├─{VBoxService}(1489)
           │                   ├─{VBoxService}(1490)
           │                   ├─{VBoxService}(1491)
           │                   ├─{VBoxService}(1492)
           │                   └─{VBoxService}(1493)
           ├─accounts-daemon(759)─┬─{accounts-daemon}(804)
           │                      └─{accounts-daemon}(846)
           ├─acpid(726)
```

systemd单元

Systemd可以管理所有的系统资源，不同的资源统称为单元。Systemd通过单元来组织和管理任务。Systemd的单元之间可以相互依赖，systemd的依赖通过以.wants为扩展名的目录来表示。它们位于/etc/systemd/system和/lib/systemd/system。其中multi-user.target.wants对应着多用户运行级别3的依赖，graphical.target.wants对应着图形界面的运行级别5。

```
lei@lei-VirtualBox:/etc/systemd/system$ ls
 bluetooth.target.wants
 cloud-final.service.wants
 dbus-fi.w1.wpa_supplicant1.service
 dbus-org.bluez.service
 dbus-org.freedesktop.Avahi.service
 dbus-org.freedesktop.ModemManager1.service
 dbus-org.freedesktop.nm-dispatcher.service
 dbus-org.freedesktop.resolve1.service
 dbus-org.freedesktop.thermald.service
 default.target.wants
 display-manager.service
 display-manager.service.wants
 final.target.wants
 getty.target.wants
 graphical.target.wants
 multi-user.target.wants
```

systemctl命令

使用systemctl命令，用户可以检查和控制systemd的状态，管理各种systemd服务。

命令格式：systemctl [options] command [name...]

选项：-t或--type：指定要列出单元的类型，多个类型用逗号隔开。

-a：列出所有的单元。

command：为systemctl提供的命令。

例如，list-units：列出当前系统中的单元。

name：为单元名称，多个用逗号分隔。

```
lei@lei-VirtualBox:/etc/systemd/system$ systemctl list-units --type=service
UNIT                                LOAD   ACTIVE SUB     DESCRIPTION
accounts-daemon.service             loaded active running Accounts Service
acpid.service                       loaded active running ACPI event daemon
alsa-restore.service                loaded active exited  Save/Restore Sound Card St
apache2.service                     loaded active running The Apache HTTP Server
apparmor.service                    loaded active exited  AppArmor initialization
apport.service                      loaded active exited  LSB: automatic crash repor
avahi-daemon.service                loaded active running Avahi mDNS/DNS-SD Stack
```

查看单元状态

使用status子命令查看systemd的运行状态

systemctl status

查看某个具体单元的状态，可以将单元名称作为参数传递给status子命令。

查看单元状态

判断单元是否在运行：

systemctl is-active mysql.service

判断单元是否启动失败：

systemctl is-failed mysql.service

判断单元是否被启用：

systemctl is-enabled mysql.service

```
lei@lei-VirtualBox:/etc/systemd/system$ systemctl is-active mysql.service
active
lei@lei-VirtualBox:/etc/systemd/system$ systemctl is-failed mysql.service
active
lei@lei-VirtualBox:/etc/systemd/system$ systemctl is-enabled mysql.service
enabled
```

单元管理

启动服务

```
sudo systemctl start mysql
```

没有指定单元的扩展名，默认为.service.

停止服务

```
sudo systemctl stop mysql
```

重启服务

```
sudo systemctl restart mysql
```

常用systemd命令

通过hostnamectl命令查看主机信息

```
lei@lei-VirtualBox:/etc/systemd/system$ hostnamectl
   Static hostname: lei-VirtualBox
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 9debc7bf8d9d4f278602158ce080b9ff
           Boot ID: a7f17463ca0d4f69b153ae6ea84ea88f
    Virtualization: oracle
  Operating System: Ubuntu 18.04.3 LTS
            Kernel: Linux 5.0.0-36-generic
      Architecture: x86-64
```

通过loginctl命令查看当前登录的用户

```
lei@lei-VirtualBox:/etc/systemd/system$ loginctl
   SESSION        UID USER             SEAT             TTY
         1       1000 lei              seat0            tty1

1 sessions listed.
```

目标

目标就是一个单元组，包含着许多功能相关的单元。启动目标，systemd就会启动目标中的所有单元。目标与传统的运行级别很相似。

可以通过list-dependencies命令查看某个目标包含的单元。

```
lei@lei-VirtualBox:/etc/systemd/system$ systemctl list-dependencies multi-user.target
multi-user.target
● ├─anacron.service
● ├─apache2.service
● ├─apport.service
● ├─avahi-daemon.service
● ├─console-setup.service
● ├─cron.service
● ├─cups-browsed.service
● ├─cups.path
```

上面列出了multi-user.target目标所包含的单元，左侧的圆点代表了单元当前的状态，绿色为已启动。

默认目标

传统的初始化程序有默认级别的选项，即系统启动时会自动进入的运行级别。Systemd提供了默认目标与之相对应。用户可以通过get-default命令获取当前默认的目标。

systemctl get-default

```
lei@lei-VirtualBox:/etc/systemd/system$ systemctl get-default
graphical.target
```

下面的命令把当前系统的默认目标修改为multi-user.target

```
lei@lei-VirtualBox:/etc/systemd/system$ systemctl set-default multi-user.target
Created symlink /etc/systemd/system/default.target → /lib/systemd/system/multi-user.target.
```

服务与守护进程管理

Linux 系统中有些程序在启动之后持续在后台运行，等待用户或其它应用程序调用，此类程序就是服务（service）。大多数服务都是通过守护进程（deamon）实现的。

例如，web服务http，文件服务nfs。

Linux守护进程按照功能分为

- 系统守护进程：系统服务，是指那些为系统本身或系统用户提供的一类服务，主要用于当前系统。
  如：作业调度服务的cron。
- 网络守护进程：网络服务，是指供客户端调用的一类服务，主要用于实现远程网络访问。
  如：web服务。

LINUX网络服务定义文件

作为网络操作系统，Linux主要用作网络服务器，提供Web、FTP、电子邮件等Internet网络服务。Linux使用Internet网络服务文件/etc/services来定义网络服务和它们对应使用的端口号及协议。每一行对应一个服务，由四个字段组成：服务名，所用端口，协议名，别名。

例如，Web服务http运行在80端口。

```
bootpc          68/tcp                          # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
finger          79/tcp
http            80/tcp          www             # WorldWideWeb HTTP
link            87/tcp          ttylink
kerberos        88/tcp          kerberos5 krb5 kerberos-sec     # Kerberos v5
kerberos        88/udp          kerberos5 krb5 kerberos-sec     # Kerberos v5
supdup          95/tcp
```

端口号的意义

Linux系统的端口号的范围为0~65535。具体分为以下三个范围

公认端口（0~1023）：用于服务和应用程序。

已注册端口（1024~49151）：分配给用户进程或应用程序。

动态或私有端口（49152~65535）：又称为临时端口，往往在连接时被动态分配给客户端应用程序。

注意：

- 实际应用中，网络服务一般通过自己的配置文件自行定义端口。
- /etc/services中定义的服务名可以作为配置文件中的参数使用。

LINUX服务启动脚本

Linux使用服务启动脚本（Service Startup Script）统一管理服务。Ubuntu将服务启动脚本保存在/etc/init.d目录下。

```
lei@lei-VirtualBox:~$ sudo ls /etc/init.d
[sudo] password for lei:
acpid             cups-browsed      networking        rsyslog
alsa-utils        dbus              network-manager   saned
anacron           dns-clean         plymouth          speech-dispatcher
apparmor          gdm3              plymouth-log      spice-vdagent
apport            grub-common       pppd-dns          udev
avahi-daemon      hwclock.sh        procps            ufw
bluetooth         irqbalance        rc                unattended-upgrades
console-setup.sh  kerneloops        rcS               uuidd
cron              keyboard-setup.sh README            whoopsie
cups              kmod              rsync             x11-common
```

实际调用的程序文件多数位于/usr/sbin目录。

服务启动脚本

Linux使用服务启动脚本统一管理服务，服务启动脚本可用于实现启动服务，重启服务，停止服务和查询服务等功能。每个服务都有相应的服务启动脚本，存放在/etc/init.d中。

例如，/etc/init.d目录下的cron文件

```
#!/bin/sh
# Start/stop the cron daemon.
#
### BEGIN INIT INFO
# Provides:          cron
# Required-Start:    $remote_fs $syslog $time
# Required-Stop:     $remote_fs $syslog $time
# Should-Start:      $network $named slapd autofs ypbind nscd nslcd winbind
# Should-Stop:       $network $named slapd autofs ypbind nscd nslcd winbind
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: Regular background program processing daemon
# Description:       cron is a standard UNIX program that runs user-specified
#                    programs at periodic scheduled times. vixie cron adds a
#                    number of features to the basic UNIX cron, including better
#                    security and more powerful configuration options.
### END INIT INFO

PATH=/bin:/usr/bin:/sbin:/usr/sbin
DESC="cron daemon"
NAME=cron
DAEMON=/usr/sbin/cron
PIDFILE=/var/run/crond.pid
                                                              1,1           Top
```

服务启动脚本与运行级别

Linux使用rc脚本统一管理每个服务的脚本程序。Ubuntu将各个运行级别对应的脚本文件存放在/etc/rc*n*.d目录中。

```
lei@lei-VirtualBox:~$ sudo ls /etc/rc
rc0.d/ rc1.d/ rc2.d/ rc3.d/ rc4.d/ rc5.d/ rc6.d/ rcS.d/
lei@lei-VirtualBox:~$ sudo ls /etc/rc0.d/
K01alsa-utils     K01gdm3         K01plymouth           K01spice-vdagent
K01avahi-daemon   K01irqbalance   K01rsyslog            K01unattended-upgrades
K01bluetooth      K01kerneloops   K01saned              K01uuidd
K01cups-browsed   K01networking   K01speech-dispatcher  README
```

/etc/rc*n*.d目录中存放的是指向/etc/init.d目录中的脚本程序的符号链接。

```
lei@lei-VirtualBox:~$ sudo ls -l /etc/rc0.d/
total 4
lrwxrwxrwx 1 root root  20 Sep 22 07:36 K01alsa-utils -> ../init.d/alsa-utils
lrwxrwxrwx 1 root root  22 Sep 22 07:36 K01avahi-daemon -> ../init.d/avahi-daemo
n
lrwxrwxrwx 1 root root  19 Sep 22 07:36 K01bluetooth -> ../init.d/bluetooth
```

服务启动脚本与运行级别

符号链接的命名规则：

- 字母S开头说明脚本是用来启动一个服务的，
- 字母K开头说明脚本是用来停止一个服务的，
- 字母S或K后面的数字表示脚本执行的顺序，
- 数字后是指向的脚本文件的名称。

执行服务启动脚本

当Linux启动或进入某运行级别时，对应脚本目录中用于启动服务的脚本将被运行，当离开该级别时用于停止服务的脚本也将自动运行，以结束在该级别中运行的这些服务。

在系统运行过程中可以也可手动实现服务的启动，重启，停止和查询等功能。基本用法如下：

/etc/init.d/服务启动脚本名 {start|stop|status|restart|reload|force_reload}

例如查询cron服务的启动状态

/etc/init.d/cron status

```
lei@lei-VirtualBox:~$ /etc/init.d/cron status
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: ena
bled)
   Active: active (running) since Tue 2019-10-01 07:42:28 EDT; 59min ago
     Docs: man:cron(8)
 Main PID: 478 (cron)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/cron.service
           └─478 /usr/sbin/cron -f
```

执行服务启动脚本

使用service命令，用法如下：

service 服务启动脚本名 {start|stop|status|restart|reload|force_reload}

在任何路径下均可使用该命令，对服务状态进行控制。

service cron status

```
lei@lei-VirtualBox:~$ service cron status
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: ena
   Active: active (running) since Tue 2019-10-01 07:42:28 EDT; 56min ago
     Docs: man:cron(8)
 Main PID: 478 (cron)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/cron.service
           └─478 /usr/sbin/cron -f
```

配置服务启动状态

安装sysv-rc-conf报错

```
lei@lei-VirtualBox:~$ sudo apt install sysv-rc-conf
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package sysv-rc-conf
lei@lei-VirtualBox:~$
```

使用vim sources.list命令 在里面sources.list 添加镜像源
deb http://archive.ubuntu.com/ubuntu/ trusty main universe restricted multiverse

```
deb http://security.ubuntu.com/ubuntu bionic-security multiverse
# deb-src http://security.ubuntu.com/ubuntu bionic-security multiverse

deb http://archive.ubuntu.com/ubuntu/ trusty main universe restricted multiverse
"/etc/apt/sources.list" 53L, 2986C                                    53,80
```

然后 sudo apt-get update

sudo apt-get install sysv-rc-conf 接着安装就可以了

管理启动脚本

Ubuntu使用update-rc.d命令管理启动脚本，修改服务的启动选项。

例如从所有运行级别中删除指定服务cups的启动项：

```
sudo update-rc.d –f cups remove
```

往所有运行级别中添加指定服务cups的启动项：

```
sudo update-rc.d –f cups defaults
```

前提是cups已在/etc/init.d目录中。

在指定运行级别中启动或关闭某项服务：

```
sudo update-rc.d [-f] <服务名> disable|enable [s|2|3|4|5]
```

例如在运行级别3和5中关闭cups服务：

```
sudo update-rc.d cups disable 3 5
```

配置服务启动状态

查看所有服务各个运行级别的启动状态

sysv-rc-conf --list

```
lei@lei-VirtualBox:~$ sysv-rc-conf --list
acpid          2:on       3:on    4:on    5:on
alsa-utils     0:off      1:off   6:off   S:on
anacron        2:on       3:on    4:on    5:on
apparmor       S:on
apport         2:on       3:on    4:on    5:on
avahi-daemon   0:off      1:off   2:on    3:on    4:on    5:on    6:off
bluetooth      0:off      1:off   2:on    3:on    4:on    5:on    6:off
cron           2:on       3:on    4:on    5:on
cups           1:off      2:on    3:on    4:on    5:on
```

启动或关闭指定服务：

sudo sysv-rc-conf 服务名 <on|off>

设置指定运行级别中服务的启动状态：

sudo sysv-rc-conf –level <运行级别列表> 服务名 <on|off>

例如设置vsftpd服务在235运行级别启动：

sudo sysv-rc-conf –level 235 vsftpd on

进程的调度启动——自动化任务配置

Linux可以将任务配置为在指定时间，时间区间，或系统负载低于特定水平时自动运行。

自动化任务通常用于指定定期备份、监控系统、运行指定脚本等工作。

1. cron用来管理周期性重复的任务调度。

- cron是一个守护进程，是一个标准的后台服务程序，cron读取系统设置来决定什么时候执行什么操作。
- 这个守护进程每分钟唤醒一次，并通过检查 crontab文件判断需要做什么。用户使用crontab命令管理 crontab文件。cron守护进程常常是在系统启动时由 init 进程启动的。

系统级任务调度

cron 使用配置文件/etc/crontab管理系统级任务调度。

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

该配置文件中并没有定义要执行的具体作业，而是在最后一个字段中设置了可执行文件的目录，分别表示每小时、每日、每周和每月要执行任务的目录。Cron守护进程使用run-parts脚本运行对应目录中的调度任务。

定义别的任务

/etc/crontab配置文件适合全局性的计划任务，每小时、每日、每周和每月要执行的任务时间点只有一个，例如每小时执行一次的任务在第17分钟执行。为计划任务指定别的时间点，可以在/etc/cron.d/目录中添加配置文件，格式同/etc/crontab，文件名可以自定义。

例如添加一个文件backup用于备份

```
#每月第1天4:10执行脚本
10 4 1 * * /root/scripts/backup.sh
```

使用CRONTAB命令

只有root用户能够通过/etc/crontab文件和/etc/cron.d目录定制cron任务调度。普通用户只能通过crontab命令创建和维护自己的cron配置文件。

- crontab命令的语法格式如下：

  crontab [-u username] file

  crontab [-u username] {-l|-r|-e}

- 第一种格式用于安装一个新的crontab 文件，安装来源就是file所指的文件。
- -u 指定具体用户的cron 文件将被修改。如果不指定，crontab将默认是当前用户。
- -l 在标准输出上显示cron文件内容。
- -r 删除用户的cron文件。
- -e 编辑用户的cron文件。当结束编辑离开时，编辑后的文件将自动安装。

使用CRONTAB命令

Crontab命令生成的cron调度文件位于/var/spool/cron/crontabs/目录，以用户名命名。

```
lei@lei-VirtualBox:/home$ sudo ls /var/spool/cron/crontabs/
lei
```

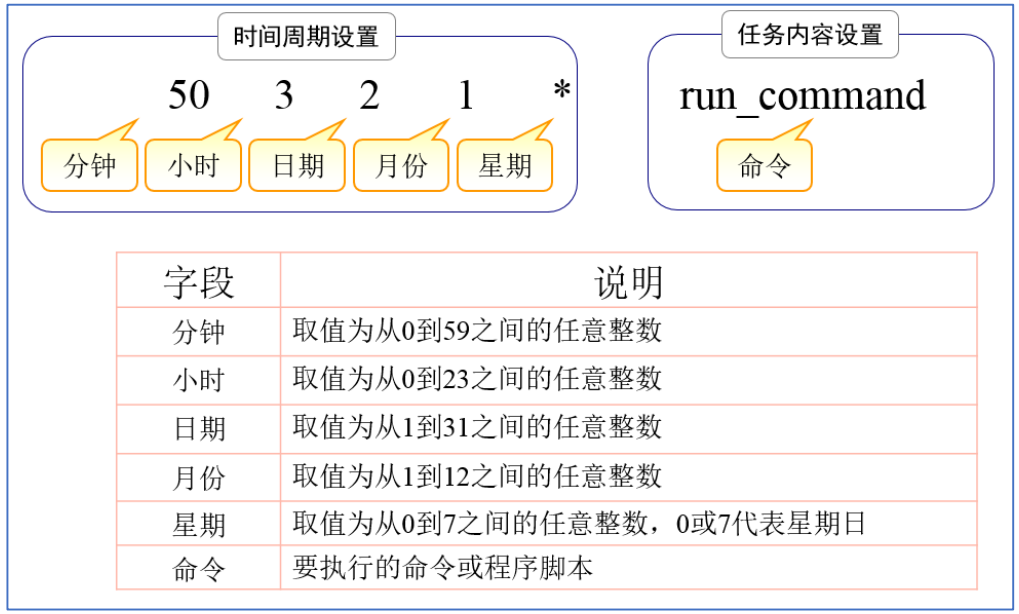
CRONTAB文件的格式

每个crontab条目包含六个字段：

分钟 小时 日 月 星期 执行的命令

- 分钟和小时的范围分别是 0-59 和 0-23，
- 日和月的范围分别是 1-31 和 1-12。
- 星期的范围是 0-6，0 表星期日。星期也可以指定为 sun、mon、tue 等等。
- 第 6 个字段包含前 5 个字段之后的所有内容，它是要传递给 sh 的字符串。

CRONTAB文件的格式



字段	说明
分钟	取值为从0到59之间的任意整数
小时	取值为从0到23之间的任意整数
日期	取值为从1到31之间的任意整数
月份	取值为从1到12之间的任意整数
星期	取值为从0到7之间的任意整数，0或7代表星期日
命令	要执行的命令或程序脚本

*　　表示该范围内的任意时间

,　　表示间隔的多个不连续时间点

-　　表示一个连续的时间范围

/　　指定间隔的时间频率

例如：

```
* * * * *   #每分钟执行一次
0 17 * * 1-5        #周一到周五每天17：00
30 8 * * 1,3,5      #每周一、三、五的8点30分
0 8-18/2 * * *      #8点到18点之间每隔2小时
0 * */3 * *         #每隔3天
```

注意：日和周不要同时使用

例子

- 5，15，25，35，45，55 16，17，18 * * * command
- 表示任意天任意月，其实就是每天的下午4点、5点、6点的5分、15分、25 分、35 分、45分、55分时执行命令。

- 0,20,40 22-23 * 7 5-6 /home/ian/mycrontest.sh
- 在 7 月的每个星期五和星期六晚上 10 点到午夜之间的第 0、20、40 分钟（每 20 分钟）执行。

使用AT安排一次性任务

在Linux中通常使用at工具在指定的时间内调度一次性任务。

安装at工具：sudo apt install at

安装后at服务自动启动， atd是其进程服务，查询其状态：/etc/init.d/atd status

```
lei@lei-VirtualBox:/home$ /etc/init.d/atd status
● atd.service - Deferred execution scheduler
   Loaded: loaded (/lib/systemd/system/atd.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-10-01 10:15:23 EDT; 2min 2s ago
     Docs: man:atd(8)
 Main PID: 2605 (atd)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/atd.service
           └─2605 /usr/sbin/atd -f
```

使用AT安排一次性任务

1. 在命令行执行at命令后跟时间参数进入作业设置状态。
2. 出现>at提示符，进入命令编辑状态，设置要执行的命令或脚本。
3. 需要结束时按<ctrl>+d退出。
4. 可执行atq查看等待运行的任务。
5. 取消作业，使用atrm后跟atq 命令输出的作业号删除作业。

```
lei@lei-VirtualBox:/home$ at now + 10minutes
warning: commands will be executed using /bin/sh
at> ps
at> ls
at> <EOT>
job 2 at Tue Oct  1 10:42:00 2019
lei@lei-VirtualBox:/home$ atq
2       Tue Oct  1 10:42:00 2019 a lei
lei@lei-VirtualBox:/home$ atrm 2
lei@lei-VirtualBox:/home$ atq
lei@lei-VirtualBox:/home$
```

时间格式

at允许使用一套相当复杂的指定时间的方法。

1）能够接受在当天的hh:mm（小时:分钟）式的时间指定。假如该时间已过去，那么就放在第二天执行。 例如：04:00

2）能够使用midnight（深夜），noon（中午），teatime（饮茶时间，一般是下午4点）等比较模糊的词语来指定时间。

3）能够采用12小时计时制，即在时间后面加上AM（上午）或PM（下午）来说明是上午还是下午。例如：12pm

4）能够指定命令执行的具体日期，指定格式为month day（月 日）或mm/dd/yy（月/日/年）或dd.mm.yy（日.月.年），指定的日期必须跟在指定时间的后面。 例如：04:00 2009-03-1

5）能够使用相对计时法。指定格式为：now + count time-units ， now就是当前时间，time-units是时间单位，这里能够是minutes（分钟）、hours（小时）、days（天）、weeks（星期）。count是时间的数量，几天，几小时。 例如：now + 5 minutes 04pm + 3 days

6）能够直接使用today（今天）、tomorrow（明天）来指定完成命令的时间。

日志管理

日志是一个必不可少的安全手段和维护系统的有力工具，实现系统审计、检测追踪、事件分析，有助于故障排除。

Ubuntu默认采用的是rsyslog，日志配置文件为/etc/rsyslog.conf。rsyslog.conf将所有配置文件放置在/etc/rsyslog.d/目录中。默认是/etc/rsyslog.d/50-default.conf，可以定制该文件实现系统日志的配置，如记录日志的信息来源，信息类型以及保存位置。

每一行代表一条设置值，格式为：信息来源.优先级　　处理方式

```
#  Default rules for rsyslog.
#
#                       For more information see rsyslog.conf(5) and /etc/rsyslog.conf

#
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
#daemon.*                       -/var/log/daemon.log
kern.*                          -/var/log/kern.log
#lpr.*                          -/var/log/lpr.log
mail.*                          -/var/log/mail.log
#user.*                         -/var/log/user.log
```

信息来源

信息来源定义日志记录来自哪个子系统。

Facility	消息来源
kern	内核消息
user	用户级消息
mail	邮件系统消息
daemon	守护进程消息
auth	认证系统消息
syslog	日志系统自身消息
lpr	打印系统消息
authpriv	权限系统消息消息
cron	定时任务消息
news	新闻系统消息
uucp	uucp系统消息
ftp	ftp服务消息

优先级

优先级代表信息的紧急程度：

debug(调试) 与none(不需登录等级)；

info：一些基本的信息；

notice：比info需要被注意到的一些信息内容；

warn：警示的讯息，可能有问题， 但是还不至于影响到某个 daemon 运作的信息；基本上， info, notice, warn 这三个讯息都是在告知一些基本信息，应该还不至于造成一些系统运作困扰；

err：一些重大的错误讯息， 例如设定文件的某些设定值造成该服务服法启动的信息说明，通常通过err的错误告知，应该可以了解到该服务无法启动的问题；

crit：比error还要严重的错误信息，这个crit是临界点(critical)的缩写，这个错误已经很严重；

alert：警告，已经很有问题的等级，比crit还要严重；

emerg：紧急，系统已经几乎要当机的状态。很严重的错误信息。通常大概只有硬件出问题，导致整个核心无法顺利运作，就会出现这样的等级的信息。

处理方式

处理方式用来定义如何处理接收到的信息，通常是将信息发往何处。

主要有以下几种处理方式：

- 将信息存储到指定文件；
- 将信息发送到指定设备；
- 将信息发给某个用户；
- 将信息发送到命名管道；
- 将信息发送到远程主机；

测试日志配置文件

保存配置文件重启系统即可生效。可以使用logger工具进行测试。例如模拟kern.info信息：

logger kern.info “test info”

logger “test log”

可以在/var/log/syslog文件中找到该日志信息。

```
Oct  1 11:17:03 lei-VirtualBox lei: kern.info test info
Oct  1 11:19:48 lei-VirtualBox lei: test log
```

查看日志内容

大部分日志文件存储在/var/log目录。日志文件中每一行就是一条信息，包含的字段主要有：信息发生的日期、时间、主机、产生信息的软件、软件或软件组件的名称、PID、信息内容。

例如，/var/log/auth.log的部分信息

```
Nov 25 14:17:01 lei-VirtualBox CRON[4394]: pam_unix(cron:session): session opene
d for user root by (uid=0)
Nov 25 14:17:02 lei-VirtualBox CRON[4394]: pam_unix(cron:session): session close
d for user root
Nov 25 14:28:33 lei-VirtualBox gdm-password]: gkr-pam: unlocked login keyring
```