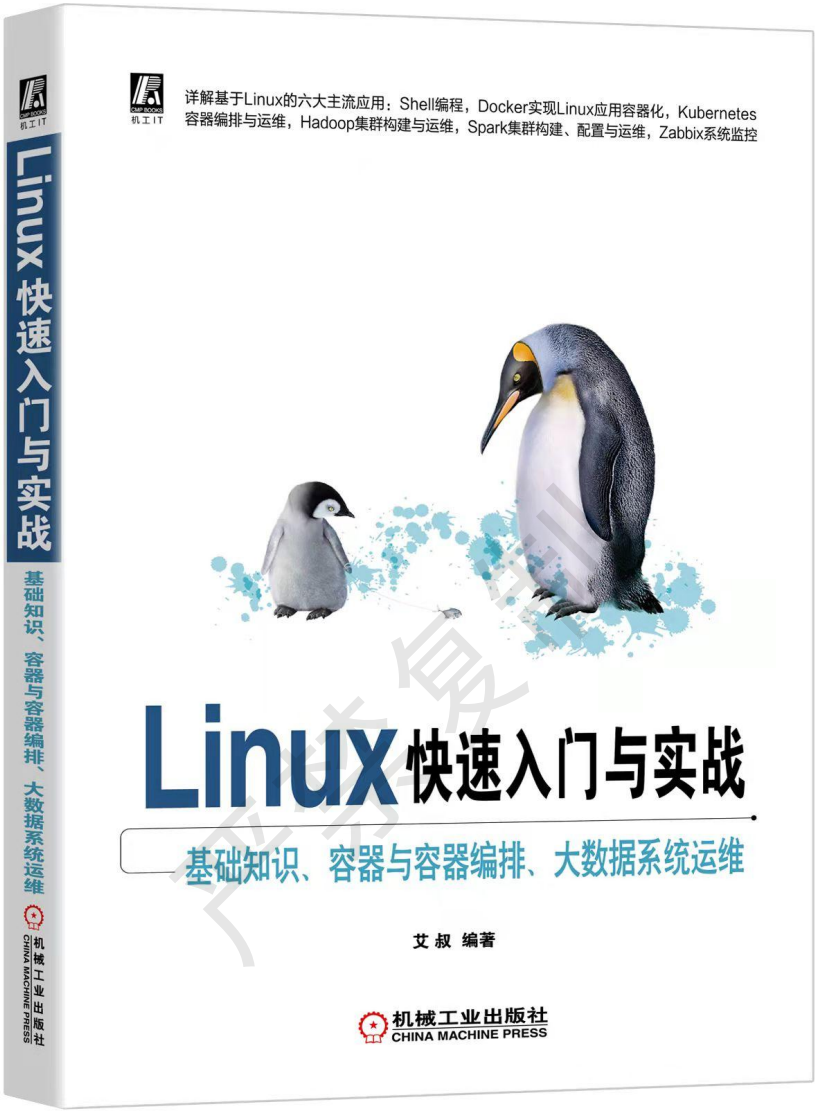
本文内容来源

《Linux 快速入门与实战 基础知识、容器与容器编排、大数据系统运维》



1、面向 Linux、大数据、云计算运维与研发工程师

2、基于 CentOS 8.2 讲解，同样适用于其他主流的 Linux 发行版

3、配套高清视频课程、资源文件、扩展阅读和实践操作电子书

4、零基础讲解 Linux 核心概念、系统组成、运行机制、高频命令和进阶使用，快速打下 Linux 坚实基础，少走弯路少踩坑

5、精选 Linux 在大数据、云原生以及系统监控等领域的实战案例，详解 Linux 下 Shell 编程、Docker、Kubernetes、Hadoop、Spark 和 Zabbix 相关技术，快速累积 Linux 实战经验

1.1 进程管理

进程指运行着的程序，进程是 Linux 系统中最基础而又重要的对象，它是一个活动的对象，有着自己的生命周期。进程的管理也是围绕着进程的生命周期开展的，包括：查看进程的谱系、前/后台进程组操作、查看进程和杀死进程等，具体说明如下。

1. 查看进程谱系

进程谱系指**进程的创建关系**。如果进程 A 创建了进程 B，那么 A 是 B 的父进程，B 是 A 的子进程，B 会继承 A 的大部分属性，如文件描述符等。同时进程 A 自身又是由另一个进程创建的，一路溯源上去，最终的源头就是 Linux 系统的初始化进程。1.2.2 节中讲过，Linux 的初始化程序第一个运行的用户态程序，CentOS 8 的初始化程序是 systemd，因此 systemd 是 CentOS 8 启动后的第一个进程，其进程 ID 为 1。

因此，查看进程谱系就是明确进程的创建关系，搞清楚进程的来龙去脉。这些信息将为读者进一步理解 Linux 系统的运行机制以及更好地使用进程提供帮助。

Linux 查看进程谱系的命令是 pstree，该命令包含在 psmisc.x86_64 包中，需要单独安装，命令如下。

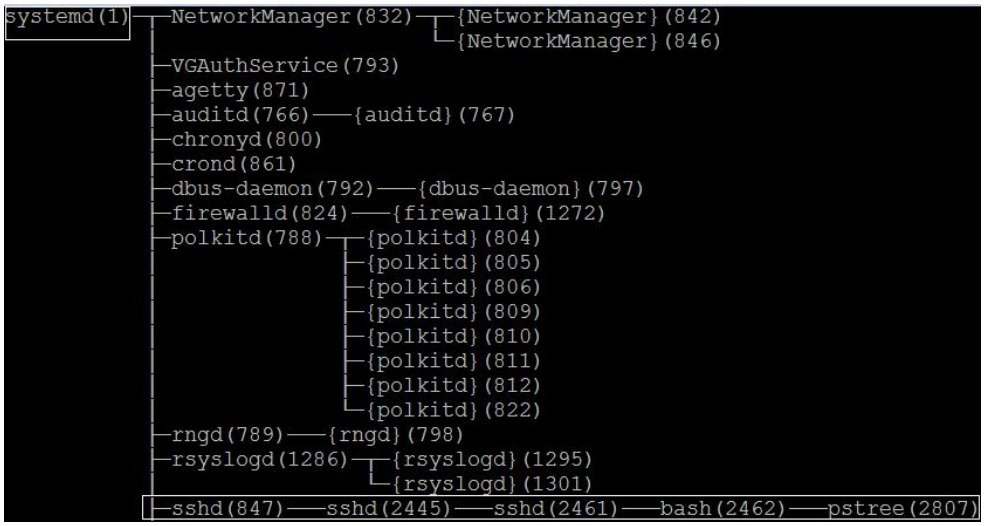
```
[root@localhost ~]# yum -y install psmisc.x86_64
```

安装后，使用“pstree -p”来显示当前系统的进程谱系，命令如下，-p 选项用于打印每个进程的 ID。

```
[root@localhost ~]# pstree -p
```

当前系统的进程谱系显示如图 1-6 所示，systemd 是所有进程的祖先，其进程 ID 为 1；刚运行的 pstree 命令也是一个进程（进程 ID 为 2807），它是在 bash 启动的，因此父进程是 bash，而 bash 的父进程又是 sshd（2461），这是因为当前操作是在 PuTTY 进行的，PuTTY 对应的服务端是 sshd，其中 sshd（847）进程是监听 SSH 服务的进程，sshd（2445）和 sshd（2461）是针对当前操作的 PuTTY 所创建的，具体创建过程可以参考 1.2 节中伪终端连接的说明。

因此进程 ID 是根据当前系统的情况分配的，因此我们每个人看到的进程 ID 会不一样。



1-6 Linux 进程谱系图

pstree 还可以显示一个进程内的线程情况，大括号内的内容为线程，例如{polkitd}就是一个线程，后面的（XXX）为线程编号，在 pstree 后加上“-t”选项可以打印线程的名字。

2. 前/后台进程组操作

Linux 系统每次 login 创建一个 Session，在这个 Session 中，会有 1 个前台进程组和多个后台进程组，有关前/台进程组的定义可以查看 1.2.4 节中说明。本节介绍前/后台进程组的常用操作示例。

（1）查看进程组信息

查看进程组信息的操作和说明如下。

1）打印 user 用户的所有的进程组信息，命令如下，因为此时 user 用户没有登录，自然不会有任何进程组信息，显示如下。

```
[root@localhost ~]# ps -u user -j
  PID  PGID   SID TTY          TIME CMD
```

上述命令和参数说明如下。

- “-u user”表示打印 user 用户的进程，如果我们要打印其他用户的进程信息，把 user 替换成其他用户的名字即可；
- -j 表示打印进程组（job）信息，具体包括：PID 是进程 ID；PGID 是进程组 ID，它取自该进程组的第一个进程的 ID，又称进程 leader ID；SID 是 Session ID，SID 取自 login 时启动的第一个进程的 ID；TTY 是该进程所关联的终端；TIME 指进程的 CPU 累积时间，不是进程的启动时刻；CMD 是启动该进程的命令。

2）在虚拟终端上以 user 用户登录。

3）在 PuTTY 上打印 user 的进程信息。命令和显示如下，由于 bash 进程组中只有 1 个进程，因此 bash 进程 ID 就是进程组 ID，因此 PGID 和 PID 一样，都是 3642。

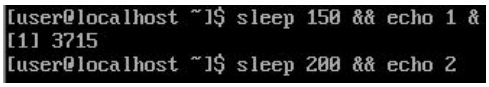
```
[root@localhost ~]# ps -u user -j
PID      PGID SID      TTY      TIME CMD
3631     3631 3631     ?        00:00:00 systemd
3636     3631 3631     ?        00:00:00 (sd-pam)
3642     3642 3642     tty1     00:00:00 bash
```

4）打印 bash（3642）进程的谱系信息，命令和显示如下，可以看到是 login 进程启动了 bash（3642），由于 bash（3642）是 login 启动的第一个进程，因此其进程 ID 3642 是此次 Session 的 ID，如上 SID 所示。

```
[root@localhost ~]# pstree -p
systemd(1)─┬─NetworkManager(832)─┬─{NetworkManager}(842)
           ├─login(3617)─────bash(3642)
```

（2）启动进程组

在当前 bash 下，再启动两个进程组，命令如图 1-7 所示。第一组命令“sleep 150 && echo 1 &”有两条命令，一个是休眠 150 秒“sleep 150”，“&&”是判断符号，如果它左边的命令“sleep 150”执行结果为 true，则执行右边的命令“echo 1”，最后的&符号，表示将该进程组放置到后台执行，因此这是一个后台进程组，它会运行一个新的 bash；第二组命令和第一组命令类似，只是没有&符号，它在前台执行，此时用户对该虚拟终端的键盘输入将会传递到该进程组的第一个进程，即 leader 进程。



1-7 进程组命令示例图

再次打印 user 用户的进程组信息，显示如下，其中 bash(3715)是和 sleep（3716）是执行第一组命令时启动的，它们属于同一个进程组，bash(3715)是该进程组的第一个进程（leader 进程），因此，该进程组的 ID 是 3715，即下面所示的 PGID 3715；sleep（3717）是第二组命令的 leader 进程，该进程组目前只有 1 个进程，即 sleep（3717），因此，它的 PGID 为 3717；此外，bash(3715)、sleep（3716）和 sleep（3717）都是在 bash（3642）中启动的，因此它们属于同一个 Session，其 Session ID 为 3642

```
[root@localhost ~]# ps -u user -j
  PID    PGID SID          TTY        TIME CMD
 3631    3631    3631      ?          00:00:00 systemd
 3636    3631    3631      ?          00:00:00  (sd-pam)
 3642    3642    3642      tty1       00:00:00 bash
 3715    3715    3642      tty1       00:00:00 bash
 3716    3715    3642      tty1       00:00:00 sleep
 3717    3717    3642      tty1       00:00:00 sleep
```

（3）进程组切换

在虚拟终端按下“Ctrl + Z”组合键，可以将前台进程组切换到后台，此时前台进程组变成 bash（3642）。

```
[user@localhost ~]$ sleep 200 && echo 2
^Z
[2]+  Stopped                 sleep 200
```

切换到后台的进程组此时是中止运行的，可以运行 bg 恢复其运行。

```
[user@localhost ~]$ bg
[2]+ sleep 200 &
```

查看当前进程组情况，命令如下，可以看到有两个进程组，进程组编号分别是 1 和 2。

```
[user@localhost ~]$ jobs
[1]-  Running                 sleep 150 && echo 1 &
[2]+  Running                 sleep 200 &
```

可以使用“fg 进程组编号”将后台进程组切换成前台进程组，示例如下，“fg 1”表示将后台进程组 1 切换成前台进程组。

```
[user@localhost ~]$ fg 1
sleep 150 && echo 1
```

2. 查看进程

查看进程的常用操作有：查看进程 ID、查看进程动态执行情况、查看进程所打开的文件等，具体示例如下。

（1）查看所有进程信息

使用“ps -eLf”可以查看当前系统所有进程的详细信息，命令如下

```
[root@localhost ~]# ps -eLf
```

上述命令和参数说明如下。

- -e 选项表示显示所有进程（包括守护进程、Session leader 进程等）；
- -L 选项和 -f 选项搭配使用将增加线程编号（LWP）和线程数量（NLWP）两列信息，这样可以查看进程内的线程信息；
- -f 选项用来显示进程的详细信息，包括 UID、PID、PPID、C、STIME、TTY、TIME 和 CMD。

上述命令的执行结果如图 1-8 所示。

```
UID          PID    PPID     LWP  C NLWP STIME TTY          TIME CMD
root           1       0       1  0    1 Nov13 ?        00:00:03 /usr/lib/systemd/systemd --switched-root --system --deserialize 17
root           2       0       2  0    1 Nov13 ?        00:00:00 [kthreadd]
root           3       2       3  0    1 Nov13 ?        00:00:00 [rcu_gp]
root           4       2       4  0    1 Nov13 ?        00:00:00 [rcu_par_gp]
root           6       2       6  0    1 Nov13 ?        00:00:00 [kworker/0:0H-kblockd]
```

1-8 进程详细信息图

上述命令显示信息的列说明如下。

- UID：用户 ID；
- PID：进程 ID；
- PPID：父进程 ID；
- LWP：线程编号，如果只有 1 个线程，那么线程编号=进程号；
- C：CPU 利用率，用整数来显示进程周期内 CPU 的百分比利用率；
- NLWP：线程数量；
- STIME：进程的起始时间；
- TTY：进程所关联的终端设备；
- TIME：CPU 累积时间；
- CMD：启动进程的命令及参数，这个参数很重要，这样我们就知道启动该进程的命令对应哪个程序，以及传入的参数是什么。

“-e”选项同“-A”，因此，“ps -e”等同于“ps -A”；

不要将“ps -A”同“ps -a”混淆起来，前者会显示系统所有的进程，后者只会显示一部分进程，其余的进程如 Session 的 leader 进程，以及不和终端关联的进程（如守护进程）都不会显示；

不要将“ps -a”同“ps a”混淆起来，前者是以标准语法来显示进程信息，后者是以 BSD 语法来显示进程信息。

为了避免上述问题，直接使用“ps -e”即可。

（2）查看进程 ID

使用 pidof 命令来查看进程的 ID。示例如下，查看 bash 进程的 ID，将显示所有 bash 进程的 ID。

```
[root@localhost ~]# pidof bash
3873 2462
```

（3）查看进程动态执行情况

使用 top 命令可以动态显示当前系统各个进程的执行情况，示例如下。

```
[user@localhost ~]$ top
```

命令执行后的显示如图 1-9 所示，经常查看的参数包括：%CPU 可以动态查看一个进程占用 CPU 资源的情况；%MEM 可以动态查看一个进程占用内存的情况。

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4655 user      20   0   63880   4356   3692 R   0.7   0.5   0:00.03 top
  794 root      20   0  152872  12660  11096 S   0.3   1.5   1:31.26 vmtoolsd
 3872 user      20   0  151112   5416   4208 S   0.3   0.7   0:02.29 sshd
```

1-9 进程动态信息图

（4）查看进程打开的文件

使用 lsof 命令来查看一个进程所打开的文件，或者是打开某个文件的所有进程。lsof 需要单独安装，命令如下。

```
[root@localhost ~]# yum -y install lsof
```

安装后，使用 lsof 查看 bash（2462）进程所打开的文件。

```
[root@localhost ~]# lsof -p 2462
```

该进程打开的文件信息如下所示，包括 stdout、stdin 和 stderr 的文件描述符。

COMMAND	PID	USER FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
bash	2462	root	cwd	DIR	253,0	173	33575041 /root
bash	2462	root	rtd	DIR	253,0	224	128 /
bash	2462	root	txt	REG	253,0	1219272	33706951 /usr/bin/bash
bash	2462	root	0u	CHR	136,0	0t0	3 /dev/pts/0
bash	2462	root	1u	CHR	136,0	0t0	3 /dev/pts/0
bash	2462	root	2u	CHR	136,0	0t0	3 /dev/pts/0

查看打开某个文件的所有进程信息，例如查看打开/dev/pts/0 文件的所有进程，命令如下，可以看到是 bash（2462）进程打开了该文件。

```
[root@localhost ~]# lsof /dev/pts/0
```

COMMAND	PID	USER FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
bash	2462	root	0u	CHR	136,0	0t0	3 /dev/pts/0
bash	2462	root	1u	CHR	136,0	0t0	3 /dev/pts/0
bash	2462	root	2u	CHR	136,0	0t0	3 /dev/pts/0
bash	2462	root	255u	CHR	136,0	0t0	3 /dev/pts/0

1.2 计划任务

计划任务可以在固定的时刻（或按照固定的周期）来执行设定的任务。这个功能在服务器领域，特别是无人值守的环境中应用很多。CentOS 8 使用 Cron 来实现计划任务，本节将对 Cron 的基本原理和典型示例进行说明。

1. Cron 基本原理

（1）crond 服务

Cron 使用 crond 服务来实现计划任务，因此，在定制任务之前，要先检查 Cron 的服务是否正常运行，命令如下。

```
[root@localhost user]# systemctl status crond
```

如果能看到 Cron 服务的状态是 running，则说明 Cron 服务运行正常。

```
   Active: active (running) since Tue 2019-11-12 18:41:18 EST; 2 days ago
```

Cron 的服务名是 crond。

（2）Cron 架构

Cron 由服务端和客户端两部分组成，服务端守护进程名字是 crond，客户端命令是 crontab。对于用户来说，使用计划任务，主要就是使用 crontab 对 Cron 进行配置。

（3）Cron 工作机制

1）每个 Linux 下的用户（root 用户，普通用户）都可以运行“crontab -e”来制定本用户的计划任务；

2）用户制定好的计划任务会存储在/var/spool/cron/目录下，以用户名作为文件名。例如 root 用户的计划任务文件名就是 root，绝对路径是/var/spool/cron/root；

3）Cron **每隔一分钟**会扫描这些配置文件的计划任务项，如果某项任务符合执行条件，就会触发该任务的执行；

Cron 扫描/触发计划任务的精度是**分钟级**的。例如，按照设置 15:12 分应该执行计划任务，但实际任务的执行时间有可能是 15:12:00、15:12:01 甚至是 15:12:02 等等。

1）Cron 的执行日志会保存在/var/log/cron 文件中。

（4）Cron 计划任务说明

一条典型的 Cron 计划任务项如下所示，分为两个部分，第一部分为条件位，由前 5 个*组成，注意*之间有空格；第二部分是 CMD，即该任务的命令及参数。

```
# * * * * * CMD
```

如上所示*表示条件位，如果条件满足，则该条件位的值为 true，否则为 false。Cron 每分钟会对所有的 Cron 计划任务项扫描一次，从左到右判断条件位，只要遇到一个条件位不为 true，就会退出，不执行 CMD 命令；如果所有条件位都为 true，则执行 CMD 命令。

条件位的含义和取值范围说明如下。

- 第一个条件位是“分钟”条件位，它表示当前时刻的分钟值，取值范围是 0~59；
- 第二个条件位是“小时”条件位，它表示当前时刻的小时值，取值范围是 0~23；
- 第三个条件位是“天数”条件位，它表示当前时刻是几号，取值范围是 1~31；
- 第四个条件位是“月份”条件位，它表示当前时刻是几月，取值范围是 1~12；
- 第五个条件位是“星期”条件位，它表示当前时刻是星期几，取值范围是 0~7，其中 0 和 7 都可以表示星期天。

每个条件位有以下几种取值，说明如下。

- 取值 1：“*”，值为 true；
- 取值 2：“n”，如果当前时刻为 n 则为 true；
- 取值 3：“m-n”，m <= 如果当前时刻 <= n，则为 true；
- 取值 4：列表形式：例如“1,2,3”，如果当前时刻为 1,2,3 中的某个值，则为 true；范围形式：例如“1-8,10-20”，如果当前时刻在 1-8 或 10-20 之内，则为 true；

取值 2、取值 3 和取值 4 可以指定在某个固定的时刻来执行计划任务，示例如下。

“1 * * * * CMD”表示每个小时的第 1 分钟执行 CMD；

“1 1 * * * CMD”表示每天的 1 点 1 分执行 CMD；

“1 1 1 * * CMD”表示每月 1 号的 1 点 1 分执行 CMD；

“* 1 * * * CMD”表示每天 1 点执行 CMD，每分钟执行 1 次；

“13-15 * * * * CMD”表示每个小时的第 13 分钟到 15 分钟执行 CMD，即 13、14 和 15 分钟各执行一次 CMD；

“13,15 * * * * CMD”表示每个小时的第 13 分钟和第 15 分钟执行 CMD。

- 取值 5：“*/n”，如果当前时刻是 n 的倍数，则为 true，注意 0 也是 n 的倍数；
- 取值 6：“m-n/k”，m <= 如果当前时刻 <= n，并且当前时间时刻是 k 的倍数，则为 true。

取值 5 和取值 6 可以指定固定的周期来执行任务，示例如下。

“*/30 * * * * CMD”表示每个小时的第 0 分钟和第 30 分钟执行 CMD；

“0-30/10 * * * * CMD”表示每个小时的第 0、10、20、30 分钟执行 CMD；

要注意各个条件位的取值范围，例如：“*/65 * * * * CMD”的第一个条件位的取值是 0~59，不可能取值 65，因此只会在每个小时的第 0 分钟执行 CMD。

2. Cron 基本使用

（1）编辑计划任务

以普通用户运行 crontab 命令，如下所示。

```
[user@localhost ~]$ crontab -e
```

每个用户都可以设置自己的计划任务，root 用户可以，普通用户也可以。要注意的是：CMD 命令是以该该用户的身份执行，要注意权限问题。

在下面的界面中编辑一条计划任务，如下所示，即每 2 分钟执行 CMD 命令，CMD 命令是“/usr/bin/date >> /tmp/c”，即将当前时刻追加到/tmp/c 文件中。

```
*/2 * * * * /usr/bin/date >> /tmp/c
```

编辑结束后，输入:wq，保存退出，如下所示。

```
:wq
```

计划任务编辑结束后，不需要重启 Cron 服务，因为 Cron 会每隔 1 分钟扫描一次计划任务配置文件，因此，该配置在下分钟扫描时就会生效。

（2）查看 Cron 执行日志

查看 Cron 执行日志，如下所示，日志路径为/var/log/cron，tail 将持续输出 cron 文件中追加的内容。

```
[root@localhost ~]# tail -f /var/log/cron
```

Cron 执行日志非常重要，我们可以通过它了解计划任务的执行情况。

Cron 日志内容如下所示，我们可以看到在 34 分、36 分都执行了 CMD。

```
Nov 16 22:34:01 localhost CROND[2009]: (user) CMD (/usr/bin/date >> /tmp/c)
Nov 16 22:36:01 localhost CROND[2018]: (user) CMD (/usr/bin/date >> /tmp/c)
```

（3）验证 Cron 执行结果

查看/tmp/c 内容，如下所示，可以看到在/tmp/c 中各有一条 34 分和 36 分的记录，说明 CMD 确实执行了。

```
[user@localhost ~]$ tail -f /tmp/c
Sat Nov 16 22:34:01 EST 2019
Sat Nov 16 22:36:01 EST 2019
```

3. Cron 注意事项

为了更好地利用 Cron，需要对 Cron 有更加深入的了解。本小节将列出 Cron 实际使用中的重要特性和注意事项，具体如下。

（1）Cron 当前计划任务的执行，不会影响后续计划任务的执行

Cron 执行计划任务时，如果当前任务还没执行完，又到了下次任务的执行时刻。此时，是等待当前任务执行完，再启动下次任务？还是依旧启动下次任务，两个任务并存呢？

答案是后一种，Cron 只负责启动任务，并不会等待任务结束。因此，Cron 每分钟依旧会扫描每个用户的每一项计划任务配置，符合条件就启动该任务；

（2）Cron 的执行条件的计算，是以时刻为依据的，而不是以 RELOAD 配置为依据

例如，某计划任务每 5 分钟执行一次 CMD，如下所示，假设 RELOAD 该配置的时间是 15:03，那么，下一次 CMD 的执行时刻是 15:05，而不是 15:08。

```
*/5 * * * * CMD
```

（3）Cron 的精度

Cron 的精度是分钟级，达不到秒级。Cron 可以做到在第几分钟执行任务，例如第 5 分钟执行任务，但是做不到第 65 分钟执行任务；Cron 可以做到每隔 7 分钟执行一次任务，但是没有办法做到所有任务都间隔 7 分钟，因为 60 无法被 7 整除（前一小时的最后一次任务，

和下一小时的第一次任务之间差的就不是 7 分钟），而且即便能整除，它们的精度也是分钟级，在秒数上可能会有差异。

（4）CMD 编写时，如果有多条命令，可以用分号 ；隔开

多条命令的 CMD 示例如下。

```
0 * * * *   /usr/bin/sleep 300;/usr/bin/echo $PATH >> /tmp/b; /usr/bin/date >> /tmp/a
```

（5）复杂 CMD 编写

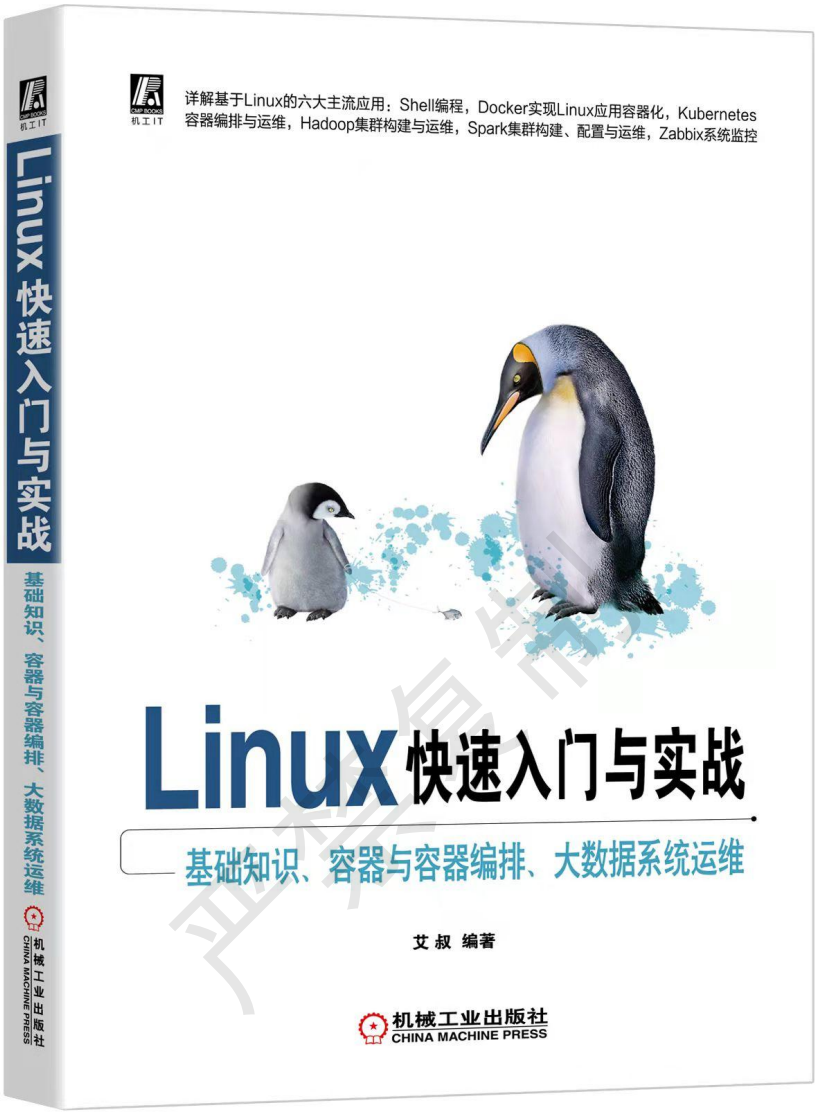
如果是复杂的 CMD，可以直接写成脚本文件，然后在 CMD 中调用。在编写脚本文件时，要注意脚本中的环境变量，例如 PATH 和 bash 中的 PATH 不一定相同，我们需要在脚本文件中，来确定这些环境变量的值，并设置它们。

4. Cron 的典型应用

在实际应用中，可以利用 Cron 来实现软件狗的功能，定期监控某个进程是否存在，如果不存在则 reboot 重启系统；还可以利用 Cron 来实现定期备份或者定期日志导出等功能。总之，凡是需要在后台定期执行的任务都可以通过 Cron 来完成。

上述内容来源

《Linux 快速入门与实战 基础知识、容器与容器编排、大数据系统运维》



1、面向 Linux、大数据、云计算运维与研发工程师

2、基于 CentOS 8.2 讲解，同样适用于其他主流的 Linux 发行版

3、配套高清视频课程、资源文件、扩展阅读和实践操作电子书

4、零基础讲解 Linux 核心概念、系统组成、运行机制、高频命令和进阶使用，快速打下 Linux 坚实基础，少走弯路少踩坑

5、精选 Linux 在大数据、云原生以及系统监控等领域的实战案例，详解 Linux 下 Shell 编程、Docker、Kubernetes、Hadoop、Spark 和 Zabbix 相关技术，快速累积 Linux 实战经验

目录

《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》

《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》

作者简介

文艾（艾叔）：原解放军理工大学-奇虎360云计算联合实验室技术负责人，系统分析师，51CTO学院严选讲师；具有多年Linux下的开发、运维和教学经验，对Linux下的Docker、Kubernetes、Hadoop和Spark等系统有深入研究和丰富的实践经验；带领团队完成了华为、中兴和奇虎360等公司的多个校企合作Linux相关项目；指导零基础本科生参加国家级科技创新竞赛和编程大赛，共获得全国特等奖1次，一等奖2次，二等奖2次；通过“艾叔编程”公众号和网易云课堂开设了一系列Linux相关的免费课程，已帮助8万多名学习者入门编程并深受好评。

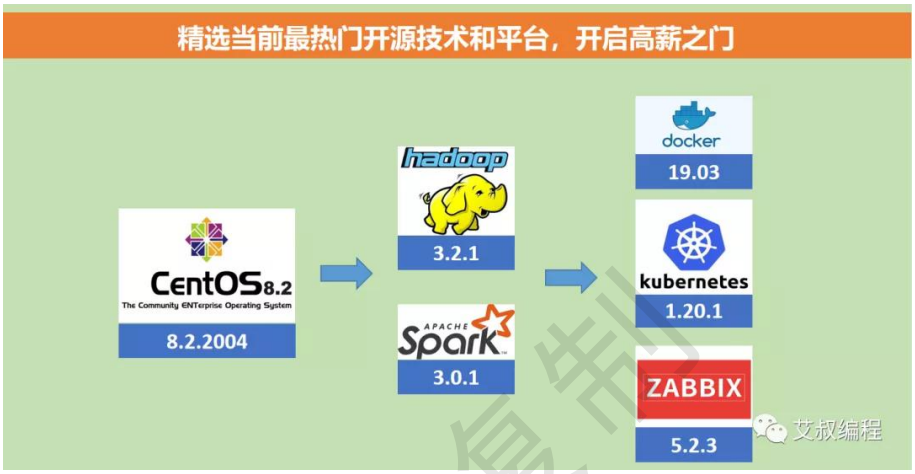
加入艾叔 Linux+大数据+云原生学习群

特色内容介绍

一本书涵盖：

Linux + Shell 编程
+ Docker + Kubernetes(K8s)
+ Hadoop(HDFS + YARN + Ozone) + Spark
+ Zabbix

涉及的都是 Linux 领域主流和热门的技术
特别是最近大火的云原生技术 Kubernetes，
则更是超越 Linux，被 Linux 基金会评为 2021 年最热门的开源技术

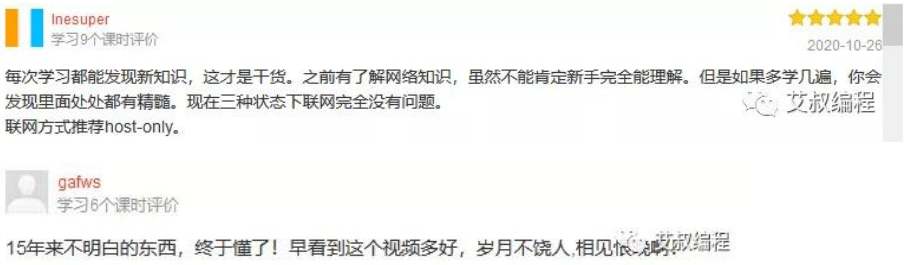


这么多技术放在一本书讲的清吗？
这个确实是一个挑战
但对艾叔来说却不难，因为，艾叔在这些领域深耕多年，深知 2/8 定律
每个技术方向，都是精挑细选，优中选优，
提炼最精华的部分浓缩成一本书。

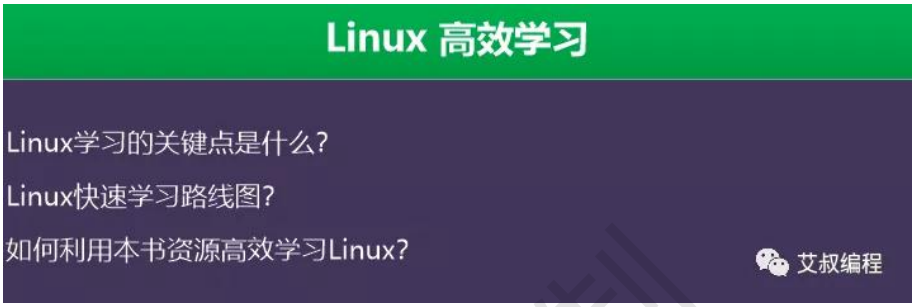
比如虚拟机内容，来源艾叔在网易云课堂上主讲的《零基础 VMware 虚拟机实战入门》的精华，这是一门五星课程，也是网易云课堂上学习人数最多，评价最好的一门课程

部分评价如下

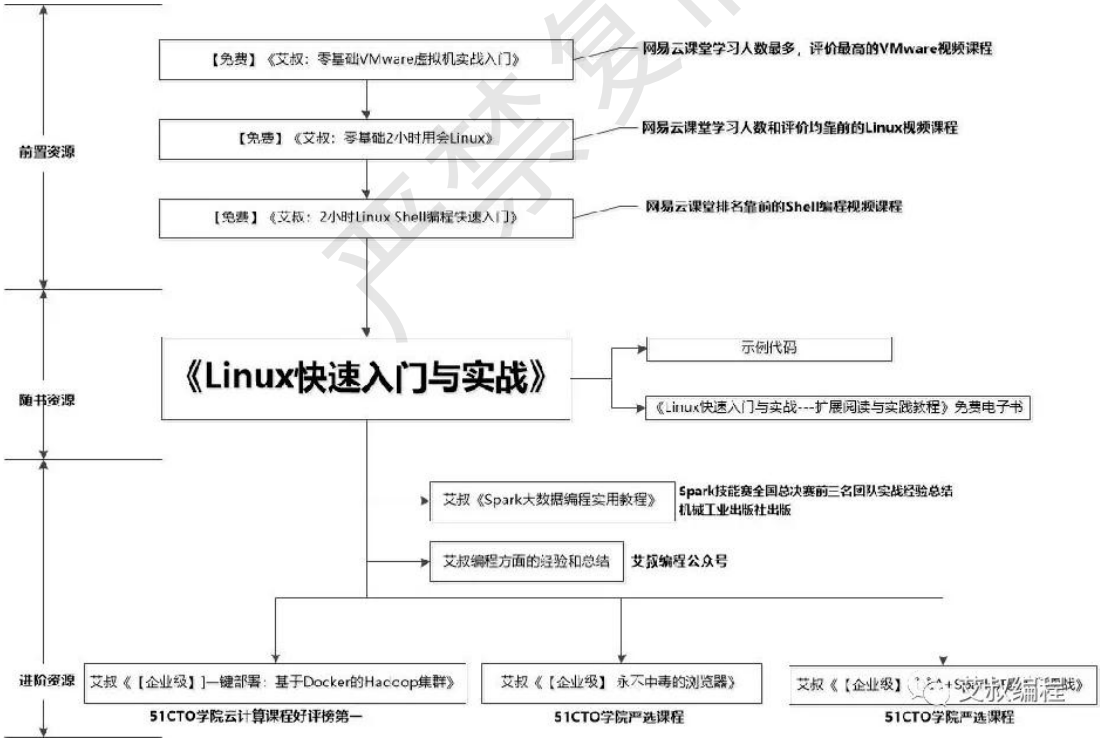
《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》



本书前面会讲 Linux 的学习方法
少走弯路，少踩坑



本书的配套资源



再讲 Linux 的核心概念，特别是容易混淆的

Linux 核心概念

Linus Torvalds为何会写Linux？他主要做了哪些工作？

Linux系统、Linux发行版、Linux内核、开源、POSIX 是什么？

Linux同Unix到底是怎样的关系？

Linux为何能逆袭Unix？

Linux、GNU、GPL、自由软件之间的关系是怎样的？

自由（free）软件就是免费软件吗？自由（free）软件和开源软件是同一个东西吗？

在Linux上开发的应用程序和内核程序一定要遵守GPL协议吗？

艾叔编程

接下来讲 Linux 的系统组成，帮助大家在头脑中构建一个静态的 Linux

Linux 系统组成

Linux系统由哪几部分组成？

为什么有的Linux系统很庞大，而有的Linux系统又很小呢？

Linux内核主要有哪些功能？

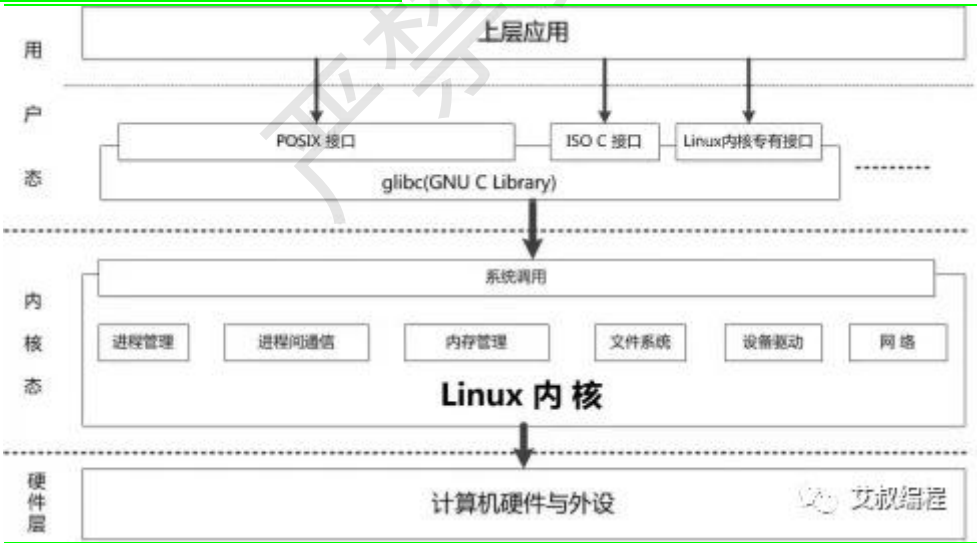
Linux内核以何种方式对外提供接口？

应用程序是如何同Linux内核打交道的呢？

什么是Glibc？Glibc是必须的吗？

root文件系统的组成有哪些？

书中总结的 Linux 内核系统架构图



再讲 Liunx 的启动、登录和交互

帮助大家在头脑中构建一个动态运行的 Linux

《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》

Linux 启动、登录和交互

Linux的启动过程，从按下开机按钮，到Linux显示登陆界面，中间经历了什么?

什么是BIOS？什么是主引导扇区？什么是主引导记录（MBR）？

我们看到的Linux启动菜单，来源于哪个程序?

Linux内核启动的大致过程?

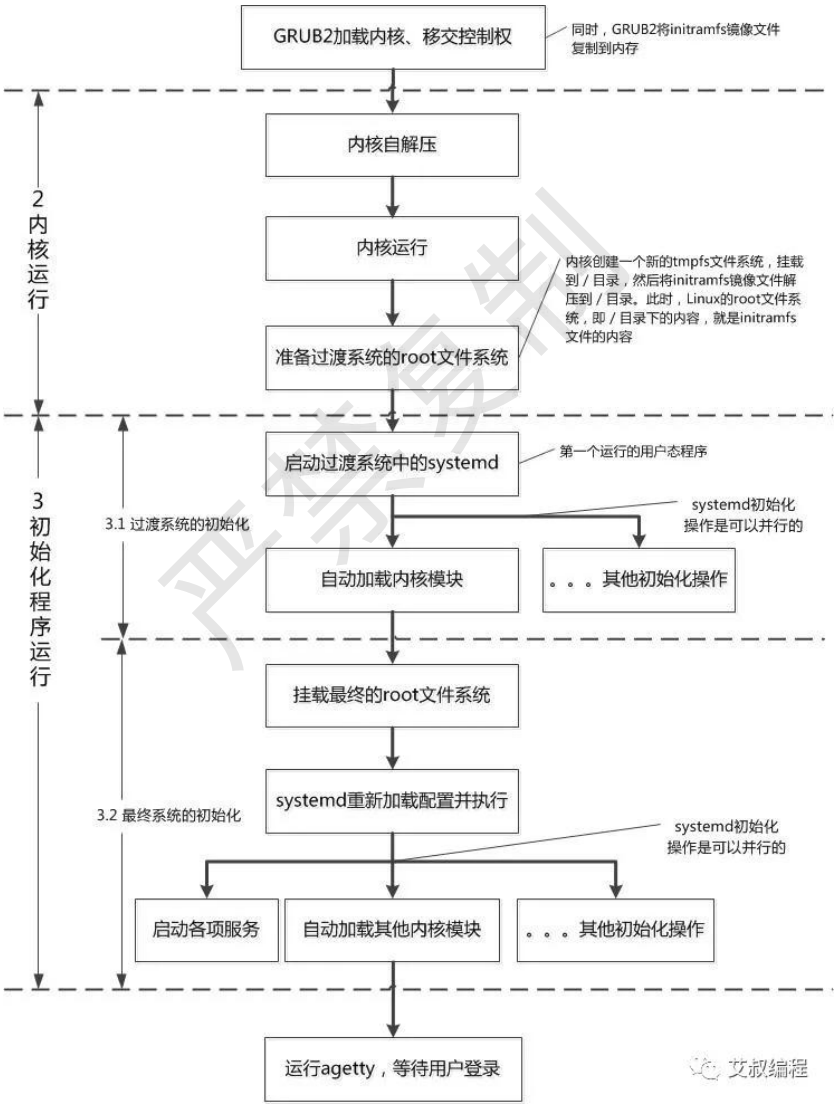
CentOS 8为什么要使用initramfs过渡系统，而不直接挂载最终的root文件系统呢?

当我们输入用户名和密码，按下Enter键后，Linux会做哪些动作?

Linux终端（虚拟终端，串行终端和伪终端）的架构是怎样的?

当我们按下按键，到Linux显示字符，这个过程经历了哪些?

书中初始化程序运行过程的细节图

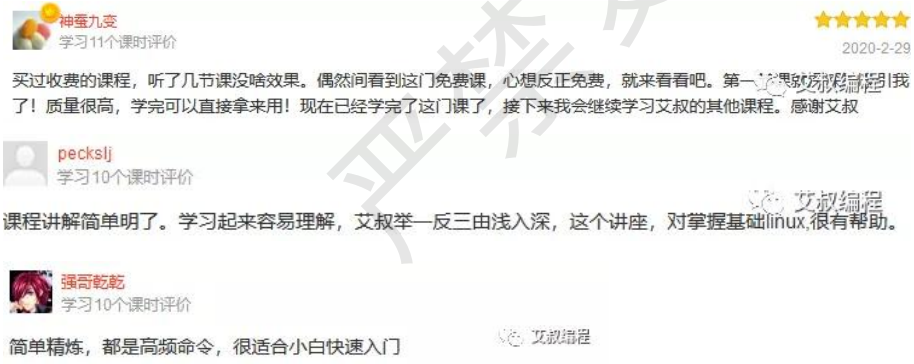


快速打下基础，进入 Linux 实操

《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》

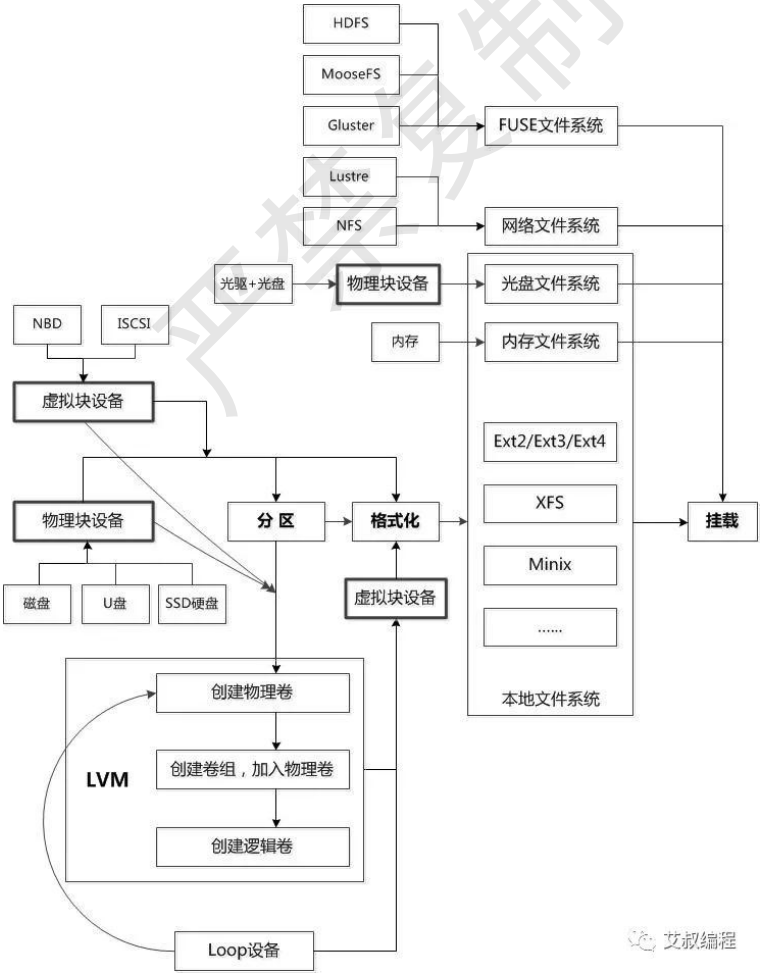
Linux 命令部分的内容
来源于艾叔在网易云课堂上主讲的《零基础 2 小时用会 Linux》的精华

这是一门 5 星课程，1.3 万同学学习后一致好评，部分评价如下



学会高频命令后，进阶学习

书中总结的 Linux 存储体系结构，对实际工作很有帮助



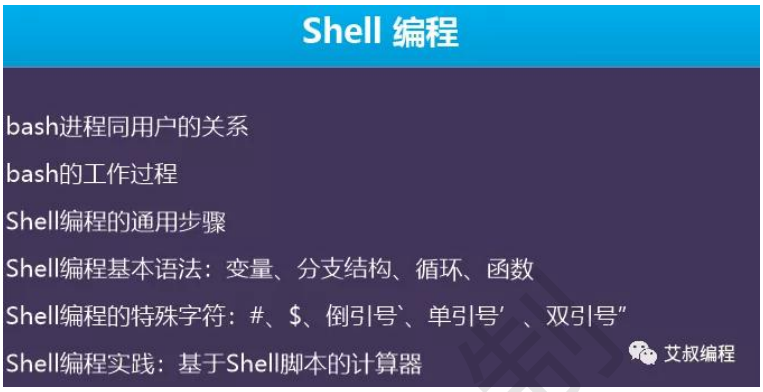
Linux 的命令学习后，深入 Linux 最好的方法，就是对 Linux 进行深度应用。因此，本书精选：**Shell+Docker+K8s+Hadoop+Spark+Zabbix** 六大应用领域。

（一）Shell 编程

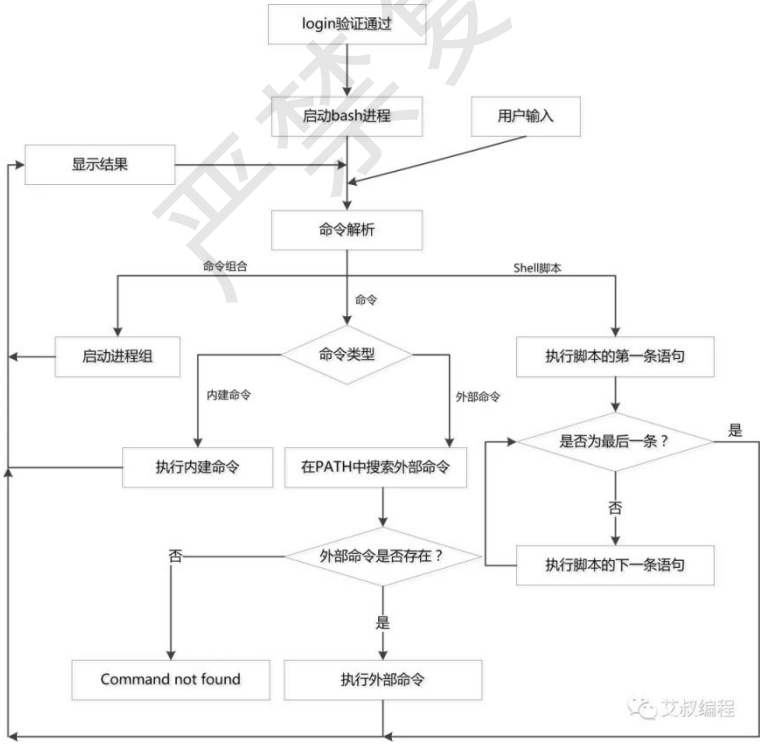
Linux 命令是最基础的执行单元，进程间通信机制（如管道）可以将命令组合成语句，而 Shell 编程则可以将语句组合成复杂的程序。因此，Shell 编程非常重要，Shell 编程语言又称为胶水语言。

Shell 编程在 Linux 系统初始化、程序安装、系统测试、运维和监控等各个领域使用非常普遍。对 Linux 用户来说，无论是从事 Linux 下的运维还是研发都需要扎实掌握 Shell 编程。

本章核心内容如下：



书中总结的 Bash 工作流程



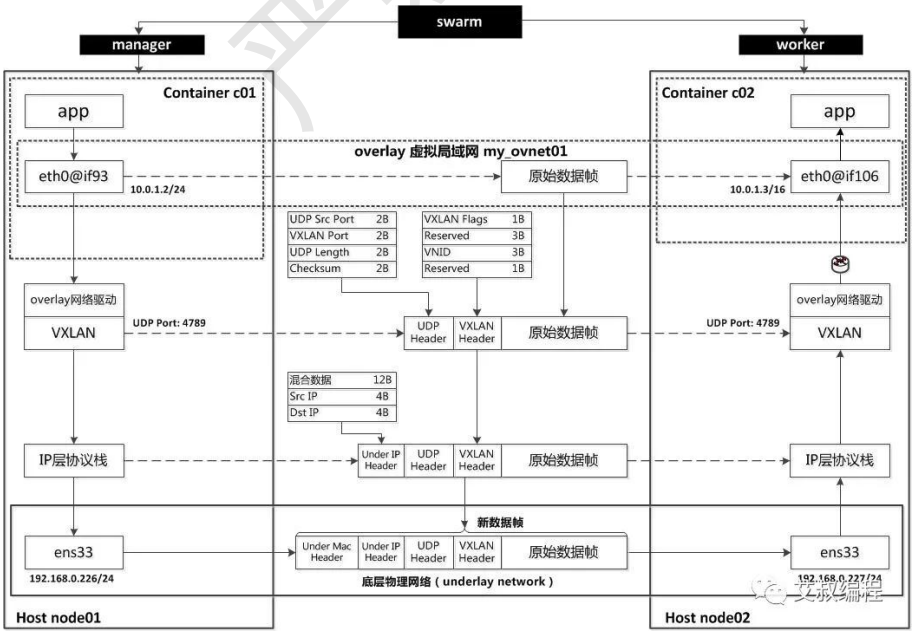
（二）Docker

Docker 是 IT 界近年来最火热的技术之一，Google、AWS 和 Microsoft 等国际巨头以及国内的阿里、腾讯和百度等，纷纷拥抱 Docker。短短几年内，无论公司大小只要涉及信息基础设施，涉及虚拟化，言必称 Docker。

Docker 出现之前，Linux 应用是和操作系统紧密耦合的，导致迁移和管理的成本很高。Docker 出现后，它将 Linux 应用的容器化，根本性地提升了 Linux 应用的开发、交付和运行效率。加上 Docker 自身在性能和易用性上的显著优势，Docker 已经深度融合到了 Linux 信息系统研发、测试、交付、部署和运维的各个环节。其影响之深，范围之广，近 10 年来都难有一种技术能与之媲美，可以说，Docker 是近年来最具影响力和颠覆性的 IT 技术之一。

本章核心内容如下：

书中总结的 Docker overlay 虚拟网络原理



本章自带的 Linux 应用的 Docker 容器化实践**《在单机上构建 100 个节点的集群》**这是一个 Linux+Shell+Docker 的综合应用，基于这种技术，我们可以很轻松的在一台虚拟机上，构建出 100 个完全分布式的节点，相对于之前使用 VMware 虚拟机，我们这个技术无论是从可行性、效率、性能、还是存储空间，等方面，都有巨大的优势。

后续，我们就可以在这个基础上，轻松构建 Hadoop、Spark 等分布式集群，非常方便。因为这些技术，我们平时在研发和工作中就是这么用的，因此，大家学习后，就可以直接应用于生产实践，非常实用。

（三）Kubernetes（K8s）

Linux 基金会报告显示，2021 年云原生技术首次超过 Linux 自身，成为当前最热门开源技术，而 Kubernetes 作为云原生技术的代表，则更是热门中的热门。因此，对于技术人员而言，Kubernets 是一个重要的加分项和加薪项。

和 Docker 相比，Kubernetes 的概念更多，更难于理解，而且 Kubernetes 是面向整个集群的容器编排，在架构、运行机制和使用上更为复杂，这些都增加了 Kubernetes 的学习难度。

因此，本章从零开始，讲解 Kubernete 技术的来龙去脉，特别是和 Docker 的关系，再从 Kubernetes 快速入门和 Kubernetes 容器编排与运维实践这 3 个方面来介绍 Kubernetes，帮助学习者快速掌握。本章核心内容如下：

本章自带 3 个综合实践，分别是：

（1）Kubernetes 容器数据的持久化存储实践

（2）Kubernetes 容器规模自动伸缩实践---HPA

（3）Kubernetes 容器服务暴露实践

以及一个 Kubernetes 运维实例：Kubernetes 故障调试实践。

（四）Hadoop

目前大数据、云计算和人工智能已成为 Linux 下研发和运维的重要就业方向，而 Hadoop 作为主流的大数据基础设施，在这些方向中的应用非常普遍。因此，对于 Linux 学习者，如果要从事大数据、云计算和人工智能这些方向的工作，掌握 Hadoop 是必备技能和加分项。Hadoop 经过 10 多年的发展，已经形成了一个庞大的生态系统。对于初学者而言，Hadoop

概念多且技术庞杂。因此，本章主要针对 Hadoop 初学者，帮助他们快速入门和累经验。本章核心内容如下：

本章自带 5 个大型综合实践，分别是：

（1）一键部署基于 Docker 的 HDFS 集群

（2）一键部署基于 Docker 的 YARN 集群

（3）一键部署基于 Docker 的 Ozone 集群

（4）HDFS HA 高可用实践

（5）HDFS 纠删码使用实践

上述实践都是 Linux+Shell+Docker+Hadoop 技术的综合应用，也是平时 Hadoop 大数据系统运维的核心技术和技能。特别是《一键部署基于 Docker 的 HDFS 集群》，国内云计算和大数据的领头羊-星环科技的核心产品 TOS 也有类似功能，当年还在中国 Hadoop 技术峰会中做了演示。

（五）Spark

Spark 是一个 Linux 下的统一的大规模数据处理分析引擎。目前，大数据、云计算和人工智能已成为 Linux 下研发和运维的重要就业方向，而 Spark 已经替代 Hadoop 的 MapReduce 框架成为事实上的通用大数据处理平台，在这些方向中的应用非常普遍。因此，对于 Linux 学习者，要从事大数据、云计算和人工智能这些方向工作的话，掌握 Spark 将成为一个必备技能和重要的加分项。本章核心内容如下：

本章内容来源于艾叔多年在 Spark 领域的经验和总结，艾叔曾指导零基础本科生参加 Spark 全国编程竞赛，战胜了多支 985 高校的研究生队，获得全国总决赛二等奖 2 次，三等奖 1 次，总分排名全国前列。

本章自带两个**综合实践：**

（1）一键部署基于 Docker 的 Spark 集群

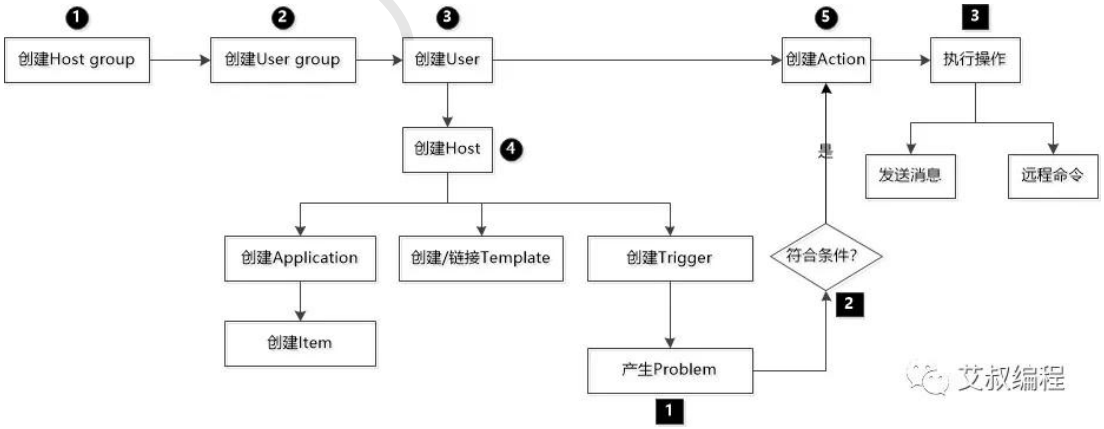
（2）基于 Docker 的 Spark History Server 构建实践

上述实践是 Linux+Shell+Docker+Spark+Hadoop 的综合应用，也是 Spark 运维和研发的核心技术。

（六）Zabbix

Zabbix 是一个 Linux 下的企业级开源监控软件，它 Linux 下系统监控的主流平台。对于 Linux 学习者，特别是有志从事 Linux 运维的学习者而言，Zabbix 是一个必备技能。本章核心内容如下：

系统监控是一个复杂工程，从硬件到平台到系统再到应用，涉及多个层次的监控对象和复杂的监控处理流程。因此，本书从小白用户的角度出发，总结了 Zabbix 使用的通用流程



本章自带大型 Zabbix 综合实战项目：Zabbix 监控 HDFS（Hadoop 分布式文件系统）此项目是 Linux+Shell+Hadoop+Docker+Zabbix 的综合应用，其中涉及多项 Zabbix 监控技术，例如：Zabbix 监控 Linux、Windows 服务器、Zabbix 监控日志、监控 MySQL 数据库、监控 Docker 容器、利用 Proxy 监控内网对象等。

《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》

图书双色印刷，纸张手感非常好👍👍👍排版也非常漂亮

是为什么 Linux 严格意义上应该称为 GNU/Linux 的原因。Linux 充分相信用户，它给用户最大的自由，它是一个可以高度定制的系统，用户可以对内核进行配置，只开启必要的内核功能，从而进一步精简内核的体积；用户也可以对周围应用进行定制，选择是否需要图形界面，选择要安装哪些应用程序。

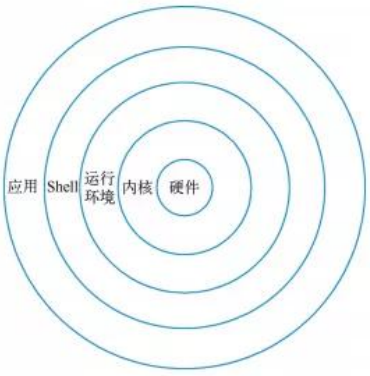


图 1-3　Linux 程序层次图

1.2.2　Linux 的启动过程

本节以 CentOS 8 启动为例，说明计算机上电后 Linux 的启动过程。了解 Linux 的启动过程有两个直接的好处：Linux 运行时出错时，可以很方便地定位是在哪个阶段出的问题；如果需要在 Linux 启动运行过程中加入自己的操作，如增加开机自启动等，可以很清楚地知道应该在哪个阶段加入。

CentOS 8 安装在 x86 架构 CPU 的计算机上，如无特殊说明，本书所涉及的计算机都是 x86 架构的计算机。

如图 1-4 所示，计算机上电后的启动顺序包括 4 个步骤：基本输入输出系统（Base Input & Output System，BIOS）自检、系统引导、内核运行和初始化程序运行。

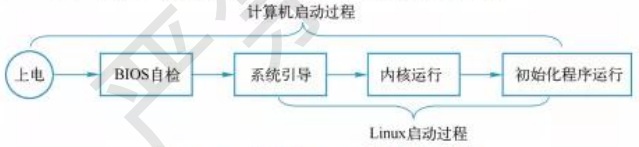


图 1-4　x86 架构下 Linux 系统启动过程图

BIOS 有两种系统引导方式，第一种是传统方式；第二种是基于统一的可扩展固件接口（Unified Extensible Firmware Interface，UEFI）方式。UEFI 相对较新，且需要主板支持。本书所使用的 VMware Workstation 9 创建的虚拟机 BIOS 中只支持传统方式，因此，本书以传统方式为例进行讲解。

钟或者设置各种外设接口等。按〈Delete〉键或者根据开机的屏幕提示信息（VMware 中是按〈F2〉键），可以进入 BIOS 设置界面。因为 BIOS 属于计算机自带程序，因此把 BIOS 自检归类到计算机启动过程，而不归类到 Linux 启动过程。

BIOS 存储在 ROM 之中，该 ROM 是可以直接被 CPU 寻址的（如 ARM 中 Norflash），因此，BIOS 程序不需要加载到内存，而是直接在 ROM 中执行。有关 BIOS 的执行过程，可以参考下面的两篇文章。

1）https://blog.csdn.net/dahuichen/article/details/53183836。

2）http://www.360doc.com/content/06/0810/13/561_177979.shtml。

Linux 启动过程如图 1-4 中所示，共分三个阶段：**系统引导、内核运行和初始化程序运行**，具体描述如下。

由于 CPU 架构或者 Linux 发行版不同，因此，系统引导和初始化程序运行阶段所采用的程序和版本可能会不同，内核运行阶段所采用的 Linux 内核版本也可能会不同。例如 CentOS 8 在系统引导阶段采用的程序是 GRUB2，而之前的 CentOS 版本采用的则可能是 GRUB1；CentOS 8 在内核运行阶段采用的 Linux 内核版本是 4.18.0，而之前的 CentOS 版本则采用更低版本的 Linux 内核；CentOS 8 在初始化程序运行阶段采用的初始化程序是 systemd，而之前的 CentOS 版本则可能采用 init 程序。

1. 系统引导

如图 1-5 所示，系统引导分为 3 个阶段：初始准备阶段，主要是读入引导程序，为引导程序的运行做准备；内核启动前的配置和管理阶段，例如提供双系统或多内核的选择菜单、配置内核启动参数和修复引导程序等；加载内核阶段，包括准备加载的环境，找到内核，最后加载内核，将控制权交给内核。

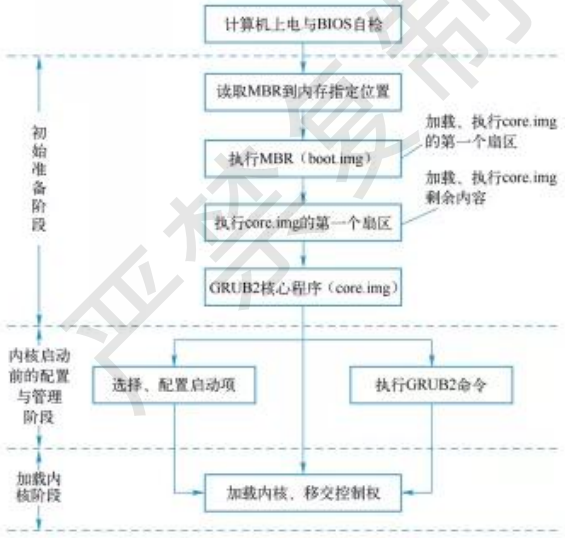


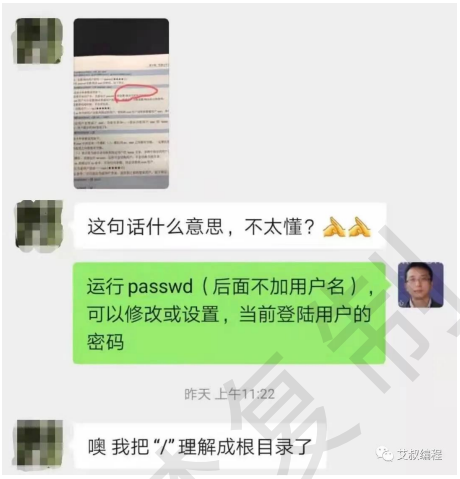
图 1-5　系统引导阶段过程

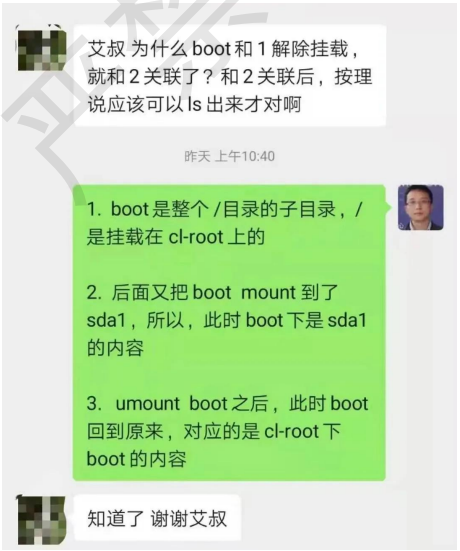
《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》

本书有专门的读者学习群

加入艾叔编程学习群

艾叔会在群中回答大家学习过程中的问题





《Linux 快速入门与实战---基础知识、容器与容器编排、大数据系统运维》