Broadview®
www.broadview.com.cn

囊括容器云的主流运维开发生态，
详细讲解基于容器云的集群运维解决方案。

容器云运维实战

——Docker与Kubernetes集群

黄靖钧　冯立灿 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

容器云运维实战

——Docker与Kubernetes集群

黄靖钧 冯立灿 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书围绕当前容器云运维的主流框架：Docker、Kubernetes 详细介绍了容器云运维的实战技巧，在内容上分为三大部分：第一部分（第 1～2 章）介绍了在 Linux 系统中传统服务器运维的基础知识以及集群管理工具；第二部分（第 3～7 章）讲解了以 Docker 为主的容器引擎的基本知识与原理，并介绍了容器技术在 DevOps 中的实际应用场景；第三部分（第 8～9 章）详细讲解了基于 Kubernetes 的容器云集群运维技巧。全书几乎囊括了容器云主流的运维开发生态，详细讲解了基于容器云的集群运维解决方案。

本书适合容器云初学者，也适合那些对 Docker 有一定了解，但对容器云的运维方式不甚了解的读者。

图书在版编目（CIP）数据

容器云运维实战：Docker 与 Kubernetes 集群 / 黄靖钧，冯立灿著. —北京：电子工业出版社，2019.3
ISBN 978-7-121-33906-6

Ⅰ. ①容… Ⅱ. ①黄… ②冯… Ⅲ. ①Linux 操作系统－程序设计 Ⅳ. ①TP316.85

中国版本图书馆 CIP 数据核字(2018)第 056964 号

策划编辑：石　倩
责任编辑：牛　勇　　特约编辑：顾慧芳
印　　刷：三河市君旺印务有限公司
装　　订：三河市君旺印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：24　字数：818 千字
版　　次：2019 年 3 月第 1 版
印　　次：2019 年 3 月第 1 次印刷
定　　价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前　言

随着 Docker 2015 年的病毒式传播和 2016 年的迅速普及应用，云计算时代的运维方式发生了很大变化。从表面上看，20 年前依靠运维工程师通过 SSH 远程连接服务器进行维护的“刀耕火种”时代早已不复存在了；但在过去的十几年里，传统集群运维工具欣欣向荣的背后依旧是 20 年前的那套远程管理方案的自动化实现，其本质不过是把重复的劳动交给计算机自动执行了。

不管是连接效率还是集群管理都不可避免地会遇到很多问题，特别是在云计算时代，数以千计的服务器集群在大中企业如同家常便饭，传统的运维手段早已黔驴技穷。而 OpenStack 的出现也只是给了 IaaS 服务商一个喘息的机会，普通企业的运维分布式集群依旧乏力。

直到虚拟化技术有了长足发展，Namespace 最后一块拼图——User Namespace 成功实现并加入 Linux Kernel 3.8，容器虚拟化技术的翘楚——LXC 终于有了与虚拟机、KVM 等技术一战高下的底气。

2014 年，Docker 一经开源便引起了业界的轰动，这个最初基于 LXC 开发的容器引擎让全世界的开发者和运维者看到了新的方向，在毫秒级的应用部署优势面前，诸多企业纷纷“倒戈”容器阵营。

随着 Google、亚马逊、微软、IBM 等云计算巨头纷纷表态并加入 OCI（Oracle 调用接口），这股容器云的浪潮在 2015 年迅速颠覆了传统的运维方案，替代它们的是一套更智能、更全面、更灵活的自动化运维体系。

在倡导“万物皆容器”的理念下，得益于容器的轻便特性，一些边缘概念也被逐渐提上了日程：微服务、Serverless、DevOps，如今的运维已不再是简单的服务器维护，更肩负了数据、服务与人的沟通。

在如今高效的集群管理方案面前，有人不禁惊呼容器时代不再需要运维工程师了，但是待我们推开容器云世界的大门时，我们发现逝去的不过是旧的运维世界，在新的容器云世界里，我们依旧知之甚少，运维工程师还不可或缺。

本书围绕当前容器云运维的主流框架：Docker、Kubernetes 详细介绍了容器云运维的实战技巧，在内容上分为三大部分：第一部分（第 1～2 章）介绍了在 Linux 系统中传统服务器运维的基础知识以及集群管理工具；第二部分（第 3～7 章）讲解了以 Docker 为主的容器引擎的基本知识与原理，并介绍了容器技术在 DevOps 中的实际应用场景；第三部分（第 8～9 章）详细讲解了基于 Kubernetes 的容器云集群运维技巧。

全书几乎囊括了容器云主流的运维开发生态，详细讲解了基于容器云的集群运维解决方案。全书内容不仅介绍了 Docker 与 Kubernetes 的基本的主流功能，还对其过渡性的实验功能和即将遗弃的

功能做了一定的提醒，对新手而言可以减少“踩坑”的概率。

因此，本书一方面可以作为面向容器云入门甚至是 Linux 入门的初级教程；另一方面，随着内容的深入与扩展，本书也适合那些对 Docker 有一定了解，但是对容器云的运维方式不甚了解的读者。本书还介绍了不同场合下对规模较大的容器的管理方案，对初创企业或者小团队的运维人员而言，也是一本不错的进阶书籍。

书中少量图片来自网络，相关代码若需要参考均在文中留有出处。由于笔者水平有限，书中存在错误或疏漏的地方在所难免，如有任何意见或建议，欢迎发邮件至 i@zuolan.me，感谢您的指正。

作　者

目　录

第1章

Linux 运维基础

说到服务器操作系统，Linux 的各种发行版以毫无悬念的姿态占领了绝大部分的市场份额。常见的桌面操作系统 Windows 与 Linux 在操作上有着很大的区别。

容器技术最初发源于 Linux，也成熟结果于 Linux，因此要掌握容器技术与云运维技术免不了要对 Linux 有一定的了解。

在推开容器云的大门之前，我们先来了解一些必备的知识。本章主要内容有：

- 了解 Linux 系统启动机制
- 掌握基本的 Linux 操作和 Shelll 编程基础
- 了解监控 Linux 系统资源的方法
- 了解传统自动化运维的方式与工具

1.1 Linux 基础

本节内容首先介绍 Linux 系统的启动过程，方便读者理解后面容器技术的概念，更容易体会到容器技术的优势。然后介绍 Linux 系统的基本知识，以便可以更轻松地掌握后面的内容。

1.1.1 systemd

计算机在启动一个操作系统时必须加载并初始化操作系统，方能运行其他的应用程序，这是计算机初始化必不可少的一个启动过程，也就是说计算机启动需要一款初始化系统。systemd 是目前 Linux 系统中最流行的初始化系统之一，能提高系统的启动效率与质量，它不仅可以让系统进程并行启动，还能很好地守护 init 进程，减少系统内存的不必要开销。

在 systemd 诞生之前，还有两个系统初始化工具，分别是 systemvinit 和 upstart，systemvinit 是一套传统的初始化系统，已经逐渐地淡出了 Linux 历史舞台，现已基本被 systemd 和 upstart 取而代之，systemd 和 upstart 有各自的特点，不过目前已经有绝大多数的 Linux 发行版都默认使用 systemd，比如 Fedora、openSUSE、Ubuntu、Gentoo、Arch Linux 等一系列 Linux 发行版。

1. systemd 基础

Linux 系统启动要执行的程序是非常多的，比如挂载文件系统、加载硬件设备、启动后台服务、激活交换分区、启动用户程序等。每一个执行任务被 systemd 称为一个配置单元（Unit）。换句话来说，挂载一个文件系统、加载硬件设备、启动系统后台服务均被称为一个配置单元。

每一个配置单元都有彼此对应的配置单元文件，比如 Apache 有对应的一个配置单元文件 apache2.service、MySQL 有对应的一个配置单元文件 msqld.service。此类型的配置单元文件的编写既简单又简洁。便于 Linux 管理人员编辑维护这些配置单元。如下是 systemd 中的一个系统日志服务的配置单元文件 syslog.service 的内容。

```
[Unit]
Description=System Logging Service   #描述信息
Requires=syslog.socket                #指定依赖
Documentation=man:rsyslogd(8)
Documentation=http://www.rsyslog.com/doc/

[Service]
Type=notify                          #服务类型
ExecStart=/usr/sbin/rsyslogd -n
StandardOutput=null
Restart=on-failure                    #指定失败时重启

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

```
Alias=syslog.service                    #别名
```

systemd 的配置单元文件可以简单分为三个部分，分别为 Uint、Service、Install。

- [Unit]区块通常是配置文件的第一个区块，用来定义 Unit 的元数据，以及配置与其他 Unit 的关系。
- [Install]通常是配置文件的最后一个区块，用来定义如何启动，以及是否开机启动。
- [Service]区块用来定义 Service 的配置，只有 Service 类型的 Unit 才有这个区块。

由于一个 Linux 操作系统有很多 systemd 的配置单元文件，并且不同的配置单元文件加载的顺序自然也是不一样的，也就是说，一个 Linux 操作系统有多个存放 systemd 配置单元的目录。以 Ubuntu 系统为例，systemd 的配置单元文件相关的位置主要有如下这些目录：

```
root@ops-admin:~# find / -name  systemd
/sys/fs/cgroup/systemd
/etc/systemd
/etc/xdg/systemd
/run/user/1000/systemd
/run/systemd
/lib/systemd
/usr/share/doc/systemd
/usr/share/systemd
/usr/lib/systemd
/var/lib/systemd
```

现在我们知道 systemd 是什么了，接下来看看 systemd 是如何工作的。

在 systemd 体系框架中，所有的服务程序都是可以并发进行启动的。比如 Avahi、D-Bus、livirtd、X11、HAL 可以同时启动。但有这样一个问题：Avahi 需要 syslog 的服务，Avahi 和 syslog 同时启动，倘若假设 Avahi 的启动比较快，syslog 还没来得及启动，然而 Avahi 却需要记录日志，在这种情况下系统就会产生问题。

因此，systemd 系统采用了各个服务之间互相依赖的解决方案，这种解决方案的相互依赖具体分成三种关系类型：socket 依赖、D-Bus 依赖以及文件系统依赖。每一种类型的依赖都可以通过相应的技术解除依赖关系，从而解决了所有服务程序并发启动冲突的问题。在 systemd 初始化系统机制中，不管程序的依赖关系如何，全部可以并行启动，若调用的服务程序存在依赖关系，则自动激活其他程序。

以 Ubuntu 系统为例，systemd 的启动顺序如下：

（1）Boot Sequence ，启动顺序，比如硬盘启动、软盘启动、U 盘启动等；

（2）Bootloader ，引导加载；

（3）kernel + initramfs(initrd)，加载内核以及 initramfs 或 initrd；

（4）rootfs，启动文件系统；

（5）/sbin/init，启动 init 进程。

对于 systemd 的所有详细启动顺序情况，我们可以使用 systemd 本身自带的 systemd-analyze，它是一个分析启动性能的工具，用于分析启动时服务时间消耗。默认显示启动是内核和用户空间的消耗时间，下面简单介绍 systemd-analyze 的基本用法。

```
# 查看启动耗时
root@ops-admin:~# systemd-analyze
# 查看每个服务的启动耗时
root@ops-admin:~# systemd-analyze blame
# 显示瀑布状的启动过程流
root@ops-admin:~# systemd-analyze critical-chain
# 显示指定服务的启动流
root@ops-admin:~# systemd-analyze critical-chain atd.service
# 将 systemd 启动顺序以及消耗时间的详细信息生成 svg
root@ops-admin:~# systemd-analyze plot > message.svg
```

以 Ubuntu 为例，执行 `systemd-analyze plot > message.svg` 命令生成矢量图（如图 1-1 所示，图中仅显示部分单元启动时间），直观地显示了 systemd 启动顺序以及消耗的时间。

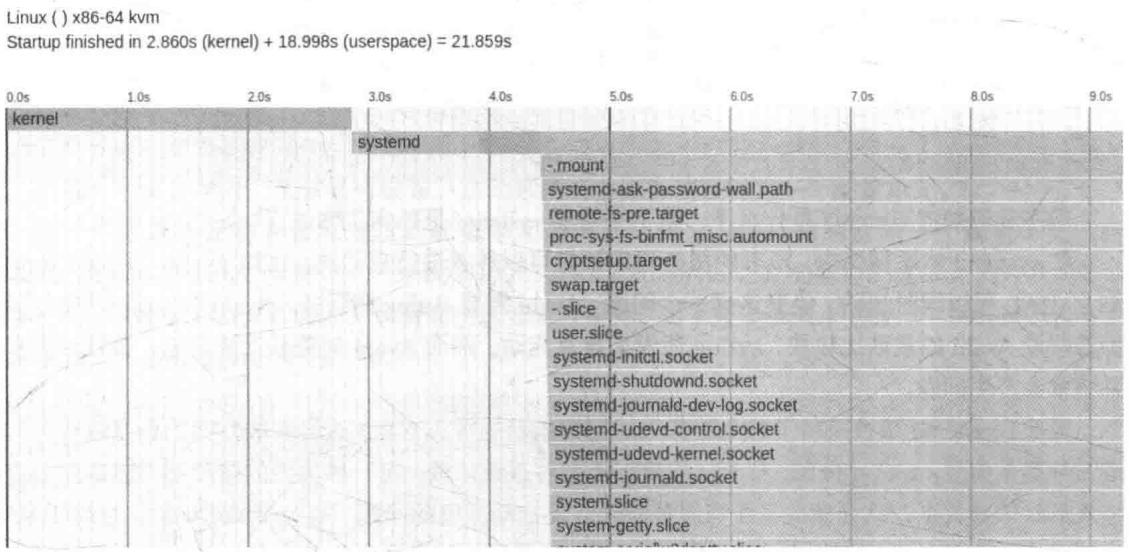

图 1-1　systemd 启动顺序及消耗的时间

2. systemctl

systemctl 是一个 systemd 工具，主要负责控制 systemd 系统和服务管理器，systemctl 工具集成了诸多的命令，使得管理员更好地管理 Linux 系统。

`systemctl list-units` 命令可以查看当前系统的所有配置单元（Unit）：

```
# 列出正在运行的 Uint
root@ops-admin:~# systemctl list-units
```

```
# 列出所有的 Uint
root@ops-admin:~# systemctl list-uints --all
# 列出所有没有运行的 Uint
root@ops-admin:~# systemctl list-units -all --state=inactive
# 列出所有加载失败的 Uint
root@ops-admin:~# systemctl list-units --failed
# 列出所有正在运行的、类型为 service 的 Unit
root@ops-admin:~# systemctl list-units --type=service
```

systemctl status 命令用于查看系统状态和每个配置单元的状态：

```
# 显示系统状态
root@ops-admin:~# systemctl status
# 显示单个 Unit 的状态，以 mysql 为例
root@ops-admin:~# systemctl status mysql.service
# 显示远程主机的某个 Unit 的状态，以 mysql 为例
root@ops-admin:~# systemctl -H {user}@{ip} status mysql.service
```

systemctl {action}对于用户来说，是常使用的命令，用于启动（start）、停止（stop）、重加载（reload）以及重启（restart）配置单元（主要是 service），下面主要以 mysql 为例：

```
# 显示 Unit 参数、启动、停止、杀死、重新加载配置等操作如下
root@ops-admin:~# systemctl <show/start/stop/restart/kill/reload> mysql.service
# 显示某个 Unit 的指定属性的值
root@ops-admin:~# systemctl show -p CPUShares mysql.service
# 重载所有修改过的配置文件
root@ops-admin:~# systemctl daemon-reload
# 设置某个 Unit 的指定属性
root@ops-admin:~# sudo systemctl set-property mysql.service CPUShares=500
```

使用 systemctl list-dependencies 命令查看 systemctl 配置单元的依赖关系，以 mysql 为例：

```
# 列出一个 mysql Unit 的所有依赖。
root@ops-admin:~# systemctl list-dependencies mysql.service
# 如果要展开 Target，就需要使用--all 参数。
root@ops-admin:~# systemctl list-dependencies --all mysql.service
```

systemctl enable/disable 命令，用于将 systemd 配置单元激活或撤销开机启动，以 mysql 为例：

```
# 激活开机启动 mysql 服务
root@ops-admin:~# systemctl enable mysql.service
# 撤销开机启动 mysql 服务
root@ops-admin:~# systemctl disable mysql.service
```

systemctl list-unit-files 命令，用于查看配置单元的所有文件（Unit File）以及配置

单元的状态（Status）情况：

```
# 列出所有配置文件
root@ops-admin:~# systemctl list-unit-files
# 列出指定类型的配置文件
root@ops-admin:~# systemctl list-unit-files --type=service
```

Unit 的状态包含如下 4 种情况。

- enabled：已建立启动链接，已激活即开机启动。
- disabled：没建立启动链接，已撤销开机启动。
- static：该配置文件没有[Install]部分，即无法被执行，只能作为其他配置文件的依赖。
- masked：该配置文件被禁止建立启动链接。

常用的 systemd 电源管理命令有如下几种：

```
# 分别为：重启机器、关机、挂起、休眠、混合休眠
root@ops-admin:~# systemctl <reboot/poweroff/suspend/hibernate/hybrid>
```

关于 Linux 电源管理的几个命令会因为发行版或者文件系统的原因有所不同，主要表现在休眠上，一些发行版不提供休眠选项，或者因为使用了不支持休眠的文件系统，例如 Btrfs 由于不支持 swap 交换分区，导致系统内存不能挂载到硬盘中，无法实现休眠。在本章中不会过多牵涉复杂的文件系统，如果没有特别指出，默认文件系统都是 ext4。

除了 systemctl 命令，还有两个命令后面也会常用。第一个是 hostnamectl 命令，它可以查看与设置主机信息：

```
# 查看主机信息
root@ops-admin:~# hostnamectl
# 设置主机信息，以设置 hostname 为例
root@ops-admin:~# hostnamectl set-hostname {$hostname}
```

第二个是 timedatectl 命令，用于管理时区：

```
# 查看时区日期等信息
root@ops-admin:~# timedatectl
# 设置主机的时区
root@ops-admin:~# timedatectl set-timezone Asia/Shanghai
# 将硬件时钟配置为地方时
root@ops-admin:~# timedatectl set-local-rtc true
```

注意：在执行上述的命令时，*.service 的后缀.service 是可以省略不写的，其执行的结果是一样的。关于更多的 systemd 内置命令请到 systemd 官方 wiki 查阅。

1.1.2 Shell 脚本

从 1.1.1 节有关 Linux 的启动过程和系统服务管理的学习中可以知道，要想在 Linux 终端环境下

行云流水地完成一系列操作，必要的命令和概念必须熟知，但这毕竟是一个积累的过程，在积累到一定量之前，还有一样必不可少的技能就是编写 Shell 脚本。Shell 脚本简单来说就是一个自动化执行命令的命令集合，但它拥有丰富的内置变量以及完善的控制语句，因此，Shell 脚本可以实现相当复杂的操作。

在学习 Shell 编程之前，本节内容会先介绍 Linux 下的权限系统，包括文件权限以及用户权限两大部分。在此之后，我们会进入基本的 Shell 编程知识讲解。在本节的最后，我们会通过几个例子，使用 Shell 脚本监控 Linux 服务器状态。

1. Linux 权限认识

你可能听说过“Linux 中一切皆文件”的说法。在认识 Linux 权限特点之前，不妨先看看 Linux 的系统目录结构，大家熟知的 Windows 目录系统结构与 Unix/Linux 大相径庭，毕竟两者在实现的机制上完全不一样。Windows 上是通过硬盘分区（C:、D:、E:）分隔目录结构的；而 Unix/Linux 系统，所有的目录结构都在一个最高级别的根目录”/“ 下，根目录是所有目录的起始点，其下面的子目录是一个层次或树状结构，这些不同的目录可以分布在不同的硬盘分区，甚至不同的设备上。

Unix/Linux 系统目录是树状目录结构，下面通过 `tree -L 1` 或者 `ls -a` 指令打印 Unix/Linux 的目录结构。了解 Linux 各个系统目录的作用，是学习 Linux 至关重要的一步，我们知道该系统的一切都是由文件组成的，并且目录结构都是由 FHS（Filesystem Hierarchy Standard）规定好了的，当我们掌握目录的结构时，管理 Linux 将会变得井然有序。下面我们具体分析根目录下的目录对应存储的文件。

- `/bin`：二进制可执行文件。
- `/boot`：系统引导文件。
- `/dev`：设备文件。
- `/etc`：系统管理和配置文件。
- `/home`：用户的家目录，基本是每一个普通用户存放一个文件夹。
- `/lib`：标准程序设计库，又叫动态链接共享库 library。
- `/lost+found`：正常的情况下，该目录为空，保存丢失文件。
- `/media`：系统会自动识别一些设备，如 U 盘、光驱等识别后，会把识别的设备挂载到这个目录下。
- `/mnt`：临时挂载别的文件系统。
- `/opt`：第三方软件默认安装的位置。
- `/proc`：虚拟的目录，它是系统内存的映射，可以通过访问这个目录来获取系统信息。
- `/root`：超级权限者的用户主目录。
- `/sbin`：即 super bin，系统管理员使用的系统管理程序。
- `/tmp`：存放一些临时文件。
- `/usr`：最庞大的目录，要用到的应用程序和文件几乎都在这个目录下。
- `/var`：某些大文件的溢出区，比方说各种服务的日志文件。

了解根目录各个文件夹的作用有助于后续遇到问题时可以正确进入相应目录查看相关信息。在 Unix/Linux 系统中，每一个资源文件仅仅属于一个所有者和一个拥有组，这样设置的目的旨在提高文件的安全性。下面我们通过 ls -la 命令来看看文件或目录的所有者与拥有组的属性。

```
user@ops-admin:~$ ls -la
total 1208
drwxrwxr-x 1 user user    532 Jul  2 22:29 Applications
drwxr-xr-x 1 user user    204 Jun 26 01:05 CodeLab
drwxr-xr-x 1 user user     20 Jul  6 21:50 Desktop
drwxrwxr-x 1 user user    348 Jun 15 20:19 Documents
drwxr-xr-x 1 user user    378 Jul  6 22:07 Downloads
-rw-rw-r-- 1 user user 1229660 Jul  6 22:16 index.pdf
drwxrwxr-x 1 user user     90 Jul  3 20:09 Share
drwxrwxr-x 1 user user    240 Jul  6 22:26 Sync
 ......
```

通过 ls -la 命令打印的信息如上，第三列的 user 就是代表文件或目录等的所有者，第四列的 user 就是代表文件或目录等的拥有组。

先看第一列的信息，第一列的信息是由 10 个字符表示的。每一位或者每一段都有其作用意义。

第 1 位：代表文件类型（文件夹或者文件），d 代表目录，-代表文件，l 代表链接文件，b 代表设备文件中可存储的接口设备，c 代表设备文件中串行端口设备，比如鼠标、键盘等。

第 2~4 位：代表文件属组权限（所有者），简称 u。

第 5~7 位：代表同组用户权限（拥有组），简称 g。

第 8~10 位：代表其他用户权限（其他用户），简称 o。

在 Unix/Linux 系统中，文件及目录的权限分为三种：可读(r|4)、可写(w|2)、可执行(e|1)，可读就是用户是否有权限查看文件的内容、可写就是用户是否由权限修改文件的内容、可执行就是用户是否具有权限执行可执行文件。就这三种权限而言，我们还应该了解文件与文件夹的权限类别，如表 1-1 所示。

表 1-1　文件与文件夹的权限类别

	文件权限	目录权限
可读	可以读取文件的内容，如：可以用 cat 命令查看文件内容	可以浏览文件目录，如：可以 cd 进入目录
可写	可以修改文件的内容，如：使用 vim 修改文件内容	可以修改目录结构，如：使用 mkdir 新建目录
可执行	可以执行可执行程序，如：执行一个可运行脚本	可以对目录执行 ls -l，并且能够 cd 进去

将上面 drwxr-xr-x 1 user user 20 Jul 6 21:50 Desktop 这一行记录作为示例讲解，第 1 位的 d 代表该文件是目录，第 2~4 位的 rwx 代表目录所有者可读可写可执行的权限，第 5~7 位的 r-x 代表目录拥有组可读可执行的权限，注意，-代表没有权限的意思，第 8~10 位的 r-x 与拥有组的权限一样，都代表可读可执行的权限。文件权限详解如图 1-2 所示。