

## 关于本书

由国内著名技术社区 51CTO 鼎力推荐、华章图书出品、2014 年南非蚂蚁（高俊峰）最新技术力作：《高性能 Linux 服务器构建实战---系统安全、故障排查、自动化运维与集群架构》，已经上架发行，此书是《高性能 Linux 服务器构建实战---运维监控、性能调优、集群应用》的姊妹篇，仍然沿用了实战、实用、通俗、易懂的写作特点，在内容上更加实战化，从运维的多个方面以近似真实的环境介绍运维工作中的各个方面，此书新增加了运维中很容易忽略但是又是非常重要的安全章节，这部分内容是本书的一大亮点。放眼同类的图书，能介绍运维安全方面的并不多；同时，此书从开始之前的简单运维阶段迈向了大规模运维内容的介绍，自动化运维篇主要介绍了海量主机的自动化部署/配置工具，接着介绍了运维的核心：自动化监控、分布式监控的应用，这些内容是对大规模集群运维下的实战介绍，这也正符合当前运维日益发展的需要。而最后的集群应用部分主要讲述了高可用集群软件、负载均衡集群软件以及 mysql 高性能集群的实例应用。纵观全局，此书对于广大 Linux 运维人员和系统管理人员来说，具有非常实用的指导意义。

本书已经在京东商城、当当网、互动网、卓越亚马逊等网上书店热销，如果你对本书感兴趣，可以在网上书店进行订购，地址如下：

京东商城：<http://item.jd.com/11533366.html>

当当网：<http://product.dangdang.com/23552973.html>

互动网：<http://product.china-pub.com/3770455>

卓越亚马逊：<http://www.amazon.cn/dp/B00N3GTWU2>

如果你对本书有疑惑，可以通过如下方式进行交流：

作者博客：<http://ixdba.blog.51cto.com>

读者交流群：8864197 134896298

作者 QQ：397824870

更多信息请访问：<http://ixdba.blog.51cto.com/2895551/1547509>

## 作者简介

高俊峰（南非蚂蚁），资深运维专家、系统架构师、DBA 和技术顾问，从事 Linux/Unix 服务器系统的架构、运维和管理多年，擅长大规模服务器集群的运维和管理，在故障诊断与排除、性能调优、自动化运维、安全运维、集群架构和虚拟化等方面积累了丰富的实战经验。国内知名 IT 技术社区 51CTO 的博客专家和 ChinaUnix 论坛 Linux 高可用板块的版主，同时还活跃于 ITPUB 等技术论坛，在社区和论坛里发表了大量技术文章，深受欢迎。此外，他还著有畅销书《循序渐进 Linux》、《高性能 Linux 服务器构建实战：运维监控、性能调优与集群应用》，后者是 Linux 运维领域公认的经典著作。

更多信息可访问：<http://baike.baidu.com/subview/2021719/11978125.htm>

## 第 5 章 Linux 故障排查案例实战

Chapter 05

### 5.1 常见系统故障案例

本章主要介绍 Linux 系统运维过程中常见的一些故障和案例，这些案例都来自实际的生产环境，每个案例都会讲述一个小问题，这些问题其实非常简单，相信大家也都遇到过。但是需要说明的是，解决问题不是本章讲述的重点，系统运维过程中的错误也千差万别，不可能把每个问题都讲述一遍，这里重点讲述的是解决问题的思路，对每个问题的讲述，我们都从错误现象开始介绍，然后是解决问题的思路，接着是问题排查过程，最后给出解决问题的方法。通过这样抽丝剥茧的介绍，最终使读者掌握解决问题的统一方法和思路，有了这样的思路，相信所有问题都能迎刃而解。

#### 5.1.1 su 切换用户带来的疑惑

这是一个客户的案例，客户的一台 Oracle 数据库服务器突然宕机了，由于在线业务的需要，客户没有考虑太多就直接重启了服务器，系统重新启动倒是没有出现问题，可是接下来，当客户准备切换到 oracle 用户下启动数据库时，怎么都无法进行 su 切换，于是问题出现了。

##### 1. 案例现象

在 root 用户下，su 切换到一个普通用户 oracle 下，却发生了错误，如图 5-1 所示。

```
[root@localhost ~]# su - oracle
su: warning: cannot change directory to /home/oracle: Permission denied
su: /bin/bash: Permission denied
```

图 5-1 su 切换发生 Permission denied 错误

于是，尝试直接通过 oracle 用户登录系统，发现此时的 oracle 用户也无法登录了，出现与上面同样的错误。

## 2. 解决思路

从上面错误提示可知是权限出现了问题，那么可以从权限入手进行排查，基本思路如下：

- ❑ 用户目录/home/oracle 权限问题
- ❑ su 程序执行权限问题
- ❑ 程序依赖的共享库权限问题
- ❑ selinux 问题导致
- ❑ 系统根空间问题

## 3. 排查问题

根据上面的思路，我们进行逐一检查，考虑到 su 在切换到 oracle 用户时会读取 oracle 目录下的环境变量配置文件，因此，首先检查/home/oracle 目录的权限是否存在问题，

```
[root@localhost home]# ls -al /home|grep oracle
drwx----- 4 oracle oinstall 4096 01-31 10:45 oracle
```

从输出可知，/home/oracle 目录的属主是 oracle 用户，而 oracle 用户对这个目录具有“rwx”权限，因此，oracle 用户目录的权限设置是正确的，可以排除掉这个问题了。

接着检查 su 执行权限问题：

```
[root@localhost home]# ll /bin/su
-rwsr-xr-x 1 root root 24120 2007-11-30 /bin/su
```

可见 su 命令执行权限也没有问题，这个也排除了。

继续检查 su 依赖的共享库权限，使用 ldd 命令检查 su 命令依赖的共享库文件，如图 5-2 所示。

```
[root@localhost home]# ldd /bin/su
linux-gate.so.1 => (0x005d3000)
libpam.so.0 => /lib/libpam.so.0 (0x00605000)
libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x005dd000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x006c8000)
libdl.so.2 => /lib/libdl.so.2 (0x00b66000)
libc.so.6 => /lib/libc.so.6 (0x00c94000)
libaudit.so.0 => /lib/libaudit.so.0 (0x00e37000)
/lib/ld-linux.so.2 (0x0065f000)
[root@localhost home]# ll /lib/ld-linux.so.2
lrwxrwxrwx 1 root root 9 2011-06-09 /lib/ld-linux.so.2 -> ld-2.5.so
[root@localhost home]# ll /lib/ld-2.5.so
-rwxr-xr-x 1 root root 129900 2011-04-27 /lib/ld-2.5.so
```

图 5-2 通过 ldd 命令检查 su 命令依赖的共享库文件

根据上面的操作，依次检查 su 命令依赖的每个库文件的权限，发现也都是正常的，都具有可执行权限，因此，共享库的问题也排除了。

根据上面的思路，继续检查 SELinux 的设置，执行命令如图 5-3 所示。

```
[root@localhost home]# grep "SELINUX" /etc/selinux/config
SELINUX=disabled
SELINUXTYPE=targeted
```

图 5-3 检查 SELinux 的设置

由输出可知，SELinux 处于关闭状态，这个原因也排除了。

到这来为止，问题变得扑朔迷离，到底是哪里出现问题了呢？作为 Linux 运维，例行检查系统根分区状态是非常必要的，那么首先检查一下根分区的磁盘空间大小，发现剩余空间还有很多，空间问题也排除。既然报的错误是权限有问题，那么只要以权限为线索，不偏离这个核心就没错，于是继续尝试检查/home 目录下各个用户的权限，执行命令如图 5-4 所示。

```
[root@localhost home]# ls -al
总计 56
drwxr-xr-x 12 root    root    4096 2012-03-30 .
drw-rw-rw- 27 root    root    4096 03-23 23:39 ..
drwx----- 3 mysql   mysql   4096 2011-03-21 mysql
drwx----- 3 nagios   nagios   4096 2011-03-26 nagios
drwx----- 4 oracle   oinstall 4096 01-31 10:45 oracle
```

图 5-4 检查/home 目录下各个用户的权限

从输出看每个用户的目录权限，都是“rwx-----”，即“700”，完全没有问题，可是我发现我错了，我的目光一直在用户对应的目录上，而忽略了其他输出信息，而问题就藏在我没有关注的信息中。在这个命令输出的前两行中，第一行权限对应的目录是“.”，代表当前目录，也就是/home 目录，权限为“drwxr-xr-x”，即“755”，第二行权限对应的目录是“..”，

也就是根目录，权限却为“rw-rw-rw-”，即“666”，此时，问题终于查找到了，原来是根目录权限问题。

将根目录权限设置为“rw-rw-rw-”，显然是不正常的。在正常情况下根目录的权限应该是“755”，为何设置成了这样，很大的可能是误操作。通过 ls 命令查看根目录的权限不是很清楚，也容易被很多运维人员忽略，其实我们可以通过另一个命令 stat 来详细查看每个目录的权限，如图 5-5 所示。

```
[root@localhost home]# stat /
File: "/"
Size: 4096          Blocks: 16          IO Block: 4096   目录
Device: 801h/2049d Inode: 2           Links: 27
Access: (0666/drw-rw-rw-)  Uid: (  0/       root)   Gid: (  0/       root)
Access: 2013-03-30 14:31:20.000000000 +0800
Modify: 2013-03-23 23:39:49.000000000 +0800
Change: 2013-03-30 14:31:17.000000000 +0800
```

图 5-5 通过 stat 命令查看根目录权限

通过这个命令，可以很清晰地看到，根目录的权限是“0666”，这才是导致 su 执行失败的根本原因。

## 4. 解决问题

知道了问题产生的原因，解决问题就非常简单，执行如下命令：

```
[root@localhost ~]#chmod 755 /
```

然后就可顺利执行 su 切换命令。

最后，做个简单的总结，这个问题主要是由于根目录没有可执行权限，而 Linux 下所有的操作都是在根目录下进行的，进而导致/home/oracle 目录没有执行权限。其实根目录权限的丢失对于系统中运行的每个用户存在同样的影响。因此，在权限出现问题时，一定要注意根目录的权限。

### 5.1.2 “Read-only file system” 错误与解决方法

这个问题经常发生在有大量磁盘读写操作且磁盘分区很大的环境中，下面简单描述下此案例的应用环境：这是一个 Web 服务器故障案例，客户利用两台服务器加一个磁盘阵列做了一个双机热备的 Web 系统，所有网站数据都存储在磁盘阵列中，两台服务器共享一个磁盘阵列分区，在正常情况下主机挂载磁盘阵列分区提供网站服务，主机故障时备机接管磁盘

阵列分区继续提供网站服务。

## 1. 案例现象

接到客户电话说他们的网站无法添加数据了，不过网站还可以正常访问，服务器和磁盘阵列也没有任何告警信息。

## 2. 解决思路

根据这个简单信息，基本排查思路如下：

- 网站程序可能出现问题了
- 服务器磁盘故障

## 3. 排查问题

首先通知研发人员对网站程序问题进行排查。经过检查，并没有发现程序有问题，而在程序日志中发现了一条信息：

```
java.lang.RuntimeException: Cannot make directory: file:/www/data/html/2013-03-30
```

根据这个输出可知，程序不能创建目录，那么尝试手动创建一个目录试试，登录 Web 服务器，在 /www/data/html 目录下创建一个目录 test，操作如下：

```
[root@localhost html]# mkdir test
```

```
mkdir: cannot create directory `test': Read-only file system
```

从这个输出信息可知，/www/data/html 目录所在的磁盘分区出现了问题，通过检查发现，/www/data/html 目录正是挂载的磁盘阵列分区，于是问题原因找到了。

## 4. 解决问题

磁盘出现“Read-only file system”的原因有很多种，可能是文件系统数据块出现不一致导致的，也有可能是磁盘故障造成的。主流的 ext3、ext4 文件系统都有很强的自我修复机制，对于简单的错误，文件系统一般可自行修复，当遇到致命错误无法修复时，文件系统为了保证数据一致性和安全，会暂时屏蔽文件系统的写操作，将文件系统变为只读，进而出现了上



面的“Read-only file system”现象。

手工修复文件系统错误的命令是 `fsck`，在修复文件系统前，最好卸载文件系统所在的磁盘分区：

```
[root@localhost ~]# umount /www/data
```

```
umount: /www/data: device is busy
```

提示无法卸载，可能这个磁盘中还有文件对应的进程在运行，检查如下：

```
[root@localhost ~]# fuser -m /dev/sdb1
```

```
/dev/sdb1:          8800
```

接着检查一下 8800 这个端口对应是什么进程，如图 5-6 所示

```
[root@localhost ~]# ps -ef|grep 8800
root      1266   1230   0 17:46 pts/0    00:00:00 grep 8800
root      8800     1   0 2012 ?        00:00:28 /usr/local/apache2/bin/httpd -k start
nobody    8801   8800   0 2012 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
nobody    8802   8800   0 2012 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
nobody    8803   8800   0 2012 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
nobody    8808   8800   0 2012 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
nobody    9106   8800   0 2012 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
nobody    27800  8800   0 Mar28 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
```

图 5-6 查询 8800 端口对应的进程信息

原来是系统的 `apache` 进程还没有停止，停止 `apache`，成功卸载磁盘，操作如下：

```
[root@localhost ~]#/usr/local/apache2/bin/apachectl stop
```

```
[root@localhost ~]# umount /www/data
```

最后，执行修复操作，如图 5-7 所示。

```
[root@localhost ~]# fsck -V -a /dev/sdb1
fsck 1.39 (29-May-2006)
[/sbin/fsck.ext3 (1) -- /www/data] fsck.ext3 -a /dev/sdb1
.....
/www/data: recovering journal
/www/data: clean, 11980860/122060800 files, 87710849/488241447 blocks
```

图 5-7 用 `fsck` 命令修复磁盘分区

修复过程比较简单，上面省略了很多输出信息。修复的时间根据磁盘大小和文件系统损坏程度而定。如果有些数据无法修复，会提示是否删除，此时可根据情况选择。修复完成后，被删除的文件会保留在对应磁盘分区挂载点的 `lost+found` 目录中。

修复完成后，执行挂载操作：

```
[root@localhost ~]# mount /dev/sdb1 /www/data
```

最后，在 `/www/data` 目录下验证是否可以成功创建文件，至此，问题圆满解决。

### 5.1.3 “Argument list too long” 错误与解决方法

作为一名运维人员，对这个错误并不陌生，在执行 `rm`、`cp`、`mv` 等命令时，如果要操作的文件数很多，可能会使用通配符批量处理大量文件，这时就可能会出现“Argument list too long”这个问题了。

#### 1. 错误现象

这是一台 MySQL 数据库服务器，在系统中运行了很多定时任务，今天通过如下 `crontab` 命令又添加了一个计划任务，退出时发生了报错。

```
#crontab -e
```

编辑完成后，保存退出，就出现如图 5-8 所示的错误。

```
crontab: installing new crontab
cron/tmp/crontab.XXXXHkEy3L: No space left on device
crontab: edits left in /tmp/crontab.XXXXHkEy3L
```

图 5-8 `crontab` 命令编辑完成后发生的错误

#### 2. 解决思路

根据上面报错的提示信息，基本判定是磁盘空间满了，那么首先从检查服务器的磁盘空间开始，先检查 `/tmp` 磁盘空间，然后检查根分区的磁盘空间，最后检查系统其他分区的磁盘空间。

#### 3. 问题排查

通过 `df` 命令查看了服务器上所有磁盘分区的情况，`/tmp` 分区空间还有很多，根分区也还有很大剩余空间，都不存在问题，最后发现是 `/var` 磁盘分区空间已经使用 100% 了。至此已经定位了问题，是 `/var` 磁盘空间爆满导致的，因为 `crontab` 会在保存时将文件信息写到 `/var` 目录下，然而这个磁盘没有空间了，所以报错也是理所当然了。



## 4. 解决问题

接着通过“du -sh”命令检查/var 目录下所有文件或目录的大小，发现/var/spool/clientmqueue 目录占用了/var 整个分区大小的 90%，那么/var/spool/clientmqueue 目录下的文件都是怎么产生的呢，是否能删除？下面简单介绍下/var/spool/clientmqueue 目录的文件是怎么生成的。

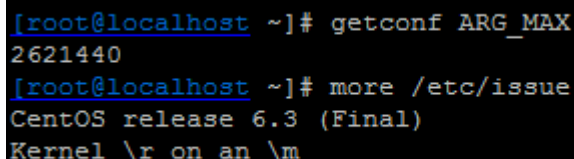
可以打开/var/spool/clientmqueue 目录下的一些文件看看，都是一些邮件信息，邮件内容大多是关于 Cron Daemon 的，其实/var/spool/clientmqueue 就是一个邮件暂存的目录。Linux 服务器在默认情况下会发一些邮件，比如当 cron 执行的程序有输出内容时，就会发送邮件信息到执行 cron 进程的用户。在发送邮件时，系统首先会把邮件复制到/var/spool/clientmqueue 目录下，然后等待 MTA (Mail Transfer Agent) 程序来处理。而 MTA 主要的功能是把这个目录中的邮件转移到/var/spool/mqueue 目录下，然后再通过 sendmail 服务发送到真正的目的地。检查这个服务器的 sendmail 服务，会发现其没有开启，这样/var/spool/clientmqueue 目录非常大的原因就找到了：没有发送邮件的客户端服务，所有邮件就都堆积在这个目录下了。

在确认完这些内容都没用后，切换到/var/spool/clientmqueue 目录下，执行 rm 命令删除所有的文件时，出现如下错误：

```
[root@localhost clientmqueue]# rm *  
/bin/rm: argument list too long
```

此时出现了本节开头我们谈到的问题。

当在 Linux 系统中试图传递太多参数给一个系统命令时，就会出现“Argument list too long”错误。这个是 Linux 系统一直以来都有的限制。查看这个限制可以通过命令“getconf ARG\_MAX”来实现，如图 5-9 所示。



```
[root@localhost ~]# getconf ARG_MAX  
2621440  
[root@localhost ~]# more /etc/issue  
CentOS release 6.3 (Final)  
Kernel \r on an \m
```

图 5-9 查看 Centos6.3 版本的 Linux 系统传递参数限制

这是 Centos6.x 版本的一个最大值，而在 Centos5.x 中，这个值相对较小，如图 5-10 所示。

```
[root@localhost ~]# getconf ARG_MAX
131072
[root@localhost ~]# more /etc/issue
CentOS release 5.8 (Final)
Kernel \r on an \m
```

图 5-10 查看 Centos5.8 版本下 Linux 下传递参数限制

所以这个问题更多是发生在 Linux 低版本中。

知道了产生问题的原因，解决方法就很多了，这里提供四种解决此问题的方法，下面分别进行介绍。

#### (1) 手动把命令行参数分成较小的部分

例如：

```
rm [a-n]* -rf
```

```
rm [o-z]* -rf
```

这种方法最简单，但是相对较弱智，因为必须要知道怎么平均分割文件，同时对于文件数目极多的情况，需要输入很多遍命令。

#### (2) 使用 find 命令删除

基本原理是通过 find 命令筛选文件列表，把符合要求的文件传递给一系列命令。这种方法是最简洁的，也是最有效的。

例如：

```
find /var/spool/clientmqueue -type f -print -exec rm -f {} \;
```

但是这种方法也有缺点：就是需要遍历所有文件，因而在文件数量极多时比较耗时。

#### (3) 通过 shell 脚本

这种方法是通过编写一个 shell 脚本，然后通过循序语句实现，与 find 方法类似。

例如，可以编写如下脚本：

```
#!/bin/bash
```

```
# 设定需要删除的文件夹
```

```
RM_DIR='/var/spool/clientmqueue'
```

```
cd $RM_DIR
```

```
for I in `ls`
```

```
do
    rm -f $I
done
```

#### (4) 重新编译 Linux 内核

这种方法需要手动增加内核中分配给命令行参数的页数，打开 kernel source 下面的 include/linux/binfmts.h 文件，找到如下行：

```
# define MAX_ARG_PAGES 32
```

将“32”改为更大的值，例如 64 或 128，然后重新编译内核。

此种方法永久有效，可以彻底解决问题，但是比较复杂，推荐给高级用户使用，没有 Linux 经验的用户不建议用这种方法。

## 5.1.4 inode 耗尽导致应用故障

### 1. 错误现象

客户的一台 Oracle 数据库服务器，在关机重启后，oracle 监听无法启动，提示错误如图 5-11 所示。

```
TNS-12549: TNSoperating system resource quota exceeded
TNS-12560: TNSrotocol adapter error
TNS-00519: Operating system resource quota exceeded
Linux Error: 28: No space left on device
```

图 5-11 inode 耗尽故障现象

从输出信息判断，应该是磁盘空间耗尽导致 Oracle 监听无法启动，因为 Oracle 在启动监听时需要创建监听日志文件，而上面三个 TNS 错误产生的原因都是由最后一行错误导致的，于是首先检查系统磁盘空间，如图 5-12 所示。

```
[www@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda6       20G   4.4G   15G   24% /
/dev/sda8       1.2T  321G  828G   28% /backup
/dev/sda5       20G   3.6G   15G   20% /usr
/dev/sda2       99G   80G   14G   80% /home
/dev/sda3       20G   16G   3.2G   84% /var
/dev/sda1       99M   18M   76M   20% /boot
tmpfs           16G    0    16G    0% /dev/shm
```

图 5-12 查看故障服务器磁盘空间信息

从磁盘输出信息可知，所有分区磁盘空间都还有不少剩余，而 oracle 监听写日志的路径在/var 分区下，虽然/var 分区仅剩下 3.2GB 可用磁盘空间，但是这对于写一个监听日志文件来说足够了，为什么还提示空间不足呢？

## 2. 解决思路

既然错误提示与磁盘空间有关，那就深入研究下关于磁盘空间的问题，在 Linux 系统中对磁盘空间的占用分为三个部分：第一个是物理磁盘空间，第二个是 inode 节点所占用的磁盘空间，第三个是 Linux 用来存放信号量的空间，而平时接触较多的是物理磁盘空间，对第二个和第三个空间的问题接触较少。既然不是物理磁盘空间的问题，接着检查是否是 inode 节点耗尽的问题，通过执行“df -i”查看系统可用的 inode 节点，如图 5-13 所示。

```
[root@localhost ~]# df -i
Filesystem          Inodes    IUsed    IFree    IUse% Mounted on
/dev/sda6            5244736  131285   5113451   3%  /
/dev/sda8            160989184 7387    160981797 1%  /backup
/dev/sda5            5244736  156624   5088112   3%  /usr
/dev/sda2            13107200 333727   12773473  3%  /home
/dev/sda3            5244736  5244736   0         100% /var
/dev/sda1            26104    41       26063     1%  /boot
tmpfs                4110598  1        4110597   1%  /dev/shm
```

图 5-13 查看磁盘分区的 inode 使用信息

由输出可知，果然是 inode 节点耗尽导致无法写日志文件。由于 inode 全部被用完了，虽然还有可用磁盘空间，但是文件系统已经无法再记录这些空余空间了，因此也就不能再创建新文件或文件夹了。由于涉及了 inode 知识，接下来就简单了解下 Linux 中 inode 的概念。

在 Linux 系统中，文件由数据块和元数据组成，数据块就是多个连续性的扇区，是文件存取的最小单位。“块(block)”的大小，最常见的是 4KB，即连续八个 sector 组成一个 block。而元数据用来记录文件的创建者、文件的创建日期、文件的大小等，这种储存文件元数据信息的区域就叫做 inode，或者称为“索引节点”。

由于 inode 也是用来存储文件相关属性信息的，因此 inode 也会消耗硬盘空间，在硬盘格式化的时候，操作系统会自动将硬盘分成两个区域。一个是数据区，存放文件数据；另一个是 inode 区 (inode table)，存放 inode 所包含的信息。

每个 inode 节点的大小，一般是 128 字节或 256 字节。inode 节点的总数在格式化文件

系统的时候，就已经确定，可以通过如下命令查看某个磁盘分区 inode 的总数：

```
[root@localhost ~]# dumpe2fs -h /dev/sda3|grep 'Inode count'
```

```
dumpe2fs 1.39 (29-May-2006)
```

```
Inode count:          5244736
```

举个形象的例子，如果将文件系统比作一本书，那么，inode 就是这本书的目录。在格式化文件系统的时候，这本书的最大目录数已经确定了。在写书（保存文件到磁盘）的过程中，可能发生这样的情况：纸用完了（磁盘空间不足），此时肯定无法保存新的文件；但是还存在另外一种情况，就是目录写完了（inode 节点全部分配完了），在这种情况下，虽然还有纸（磁盘空间），但是目录（inode）已经没有了，在文件系统上当然不能新建文件了，因为没有了目录，文件就无法通过索引找到。

另外，每个 inode 都有一个号码，操作系统用 inode 号码来区分不同的文件。通过“ls -li”命令，可以查看文件名对应的 inode 号，例如：

```
[root@localhost ~]# ls -li install.log
```

```
325762 install.log
```

如果要查看这个文件更详细的 inode 信息，可以通过 stat 命令实现，如图 5-14 所示。

```
[root@localhost ~]# stat install.log
File: `install.log'
Size: 39224          Blocks: 88          IO Block: 4096    regular file
Device: 806h/2054d  Inode: 325762       Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)  Gid: (  0/   root)
Access: 2011-07-26 19:09:02.000000000 +0800
Modify: 2011-07-26 19:17:38.000000000 +0800
Change: 2011-07-26 19:17:43.000000000 +0800
```

图 5-14 查看某文件 inode 的详细信息

### 3. 解决问题

知道了产生这个故障是 inode 导致的后，接下来就要查看/var 目录下为何耗尽了 inode，通过检查发现/var/spool/clientmqueue/这个目录里面的文件仅 500 多万个，至于产生的原因，分析后确定应该是系统的 crontab 导致的，因为系统开了多个 crontab 任务，而如果 crontab 任务没有重定向，默认就会在这个目录下创建一个文件，日积月累，此目录下的小文件就会超多。解决的方法很简单，删除这些没用的文件即可删除的方法是直接使用 rm 命令，这时肯定会提示“Argument list too long”的错误，而解决这个的方法上面章节已经有过详细介绍。

绍，通过如下命令即可完成：

```
[root@localhost ~]# find /var/spool/clientmqueue/ -name "*" -exec rm -rf {} \;
```

删除日志文件后，再次启动 Oracle 监听，可以顺利实现启动，查看新的监听日志文件已经生成，至此，问题得到圆满解决。

## 5.1.5 文件已删除但空间不释放的原因

### 1. 错误现象

运维的监控系统发来通知，报告一台服务器空间满了，登录服务器查看，根分区确实没有空间了，如图 5-15 所示。

```
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       97G   97G    0 100% /
/dev/sda1       99M   18M   76M  20% /boot
tmpfs           16G    0   16G   0% /dev/shm
```

图 5-15 查看服务器磁盘空间

这里首先说明一下服务器的一些删除策略，由于 Linux 没有回收站功能，我们将线上服务器上所有要删除的文件都会先移动到系统/tmp 目录下，然后定期清除/tmp 目录下的数据。这个策略本身没有问题，但是通过检查发现这台服务器的系统分区中并没有单独划分/tmp 分区，这样/tmp 下的数据其实占用了根分区的空间。既然找到了问题，那么删除/tmp 目录下一些大数据即可，检查/tmp 下最大的三个数据文件，如图 5-16 所示。

```
[root@localhost ~]# du -sh /tmp/*|sort -nr|head -3
66G    /tmp/access_log
36K    /tmp/hsperfdata_root
36K    /tmp/hsperfdata_mapred
```

图 5-16 查看/tmp 下前三个最大的数据文件

通过命令输出发现在/tmp 目录下有个 66GB 大小的文件 access\_log，这个文件应该是 apache 产生的访问日志文件，从日志大小来看，应该是很久没有清理 apache 日志文件了，基本判定是这个文件导致的根空间爆满，在确认此文件可以删除后，执行如下删除操作：

```
[root@localhost ~]# rm /tmp/access_log
```

接着查看系统根分区空间是否释放，如图 5-17 所示。



```
[root@localhost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       97G   97G    0 100% /
/dev/sda1       99M   18M   76M  20% /boot
tmpfs           16G    0   16G   0% /dev/shm
```

图 5-17 查看磁盘空间是否释放

从输出可以看到，根分区空间仍然没有释放，这是怎么回事？

## 2. 解决思路

一般说来不会出现删除文件后空间不释放的情况，但是也存在例外，比如文件被进程锁定，或者有进程一直在向这个文件写数据等，要理解这个问题，就需要知道 Linux 下文件的存储机制和存储结构。

一个文件在文件系统中的存放分为两个部分：数据部分和指针部分，指针位于文件系统的 meta-data 中，在将数据删除后，这个指针就从 meta-data 中清除了，而数据部分存储在磁盘中。在将数据对应的指针从 meta-data 中清除后，文件数据部分占用的空间就可以被覆盖并写入新的内容，之所以在出现删除 access\_log 文件后，空间还没释放，就是因为 httpd 进程还在一直向这个文件写入内容，导致虽然删除了 access\_log 文件，但是由于进程锁定，文件对应的指针部分并未从 meta-data 中清除，而由于指针并未被删除，系统内核就认为文件并未被删除，因此通过 df 命令查询空间并未释放也就不足为奇了。

## 3. 问题排查

既然有了解决问题的思路，那么接下来看看是否有进程一直在向 access\_log 文件中写数据，这里需要用到 Linux 下的 lsof 命令，通过这个命令可以获取一个已经被删除但仍然被应用程序占用的文件列表，命令执行如图 5-18 所示。

```
[root@localhost ~]# lsof | grep delete
mysqld  3362  mysql  4u    REG    253,0  0  75694090 /tmp/ibr0ZPnv (deleted)
mysqld  3362  mysql  5u    REG    253,0  0  75694091 /tmp/ibcPLKkn (deleted)
mysqld  3362  mysql  6u    REG    253,0  0  75694092 /tmp/ib8nXFhf (deleted)
mysqld  3362  mysql  7u    REG    253,0  0  75694093 /tmp/ibU3bHf7 (deleted)
mysqld  3362  mysql  11u   REG    253,0  0  75694095 /tmp/ibvfslpZ (deleted)
httpd   2986  apache 3w    REG    8  70866960384 75694218 /tmp/aces.log (deleted)
```

图 5-18 查看被应用程序锁定的删除文件列表

从输出结果可以看到，/tmp/aces.log 文件被进程 httpd 锁定，而 httpd 进程还一直向这

个文件写入日志数据。从第七列可知，这个日志文件大小仅 70GB，而系统根分区总大小才 100GB，由此可知，这个文件就是导致系统根分区空间耗尽的罪魁祸首。最后一列的“deleted”状态说明这个日志文件已经被删除，但由于进程还在一直向此文件写入数据，空间并未释放。

## 4. 解决问题

到这里问题就基本排查清楚了，解决这一类问题的方法有很多种，最简单的方法是关闭或重启 httpd 进程，当然也可以重启操作系统，不过这并不是最好的方法。对待这种进程不停对文件写日志的操作，要释放文件占用的磁盘空间，最好的方法是在线清空这个文件，具体可以通过如下命令完成：

```
[root@localhost ~]# echo "" >/tmp/access_log
```

通过这种方法，磁盘空间不但可以马上释放，也可保障进程继续向文件写入日志，这种方法经常用于在线清理 Apache、Tomcat、Nginx 等 Web 服务产生的日志文件。

### 5.1.6 “Too many open files” 错误与解决方法

#### 1. 问题现象

这是一个基于 Java 的 Web 应用系统，在后台添加数据时提示无法添加，于是登录服务器查看 tomcat 日志，发现了如下异常信息：

```
java.io.IOException: Too many open files
```

通过这个错误，基本判断是系统可用的文件描述符不够了，由于 tomcat 服务是系统 www 用户启动的，于是以 www 用户登录系统，通过“ulimit -n”命令查看系统可以打开最大文件描述符的数量，输出如下：

```
[www@tomcatserver ~]$ ulimit -n  
65535
```

可以看到这个服务器设置的最大可打开的文件描述符已经是 65535 了，这么大的值应该够用了，但是为什么是提示这样的错误呢？

## 2. 解决思路

这个案例涉及 Linux 下 `ulimit` 命令的使用，这里简单介绍下 `ulimit` 的作用和使用技巧。

`ulimit` 主要是用来限制进程对资源的使用情况的，它支持各种类型的限制，常用的有：

- 内核文件的大小限制
- 进程数据块的大小限制
- Shell 进程创建文件大小限制
- 可加锁内存大小限制
- 常驻内存集的大小限制
- 打开文件句柄数限制
- 分配堆栈的最大大小限制
- CPU 占用时间限制用户最大可用的进程数限制
- Shell 进程所能使用的最大虚拟内存限制

`ulimit` 使用的基本格式为：

`ulimit [options] [limit]`

具体的 `ulimit` 参数(options) 含义如表 5.1 所示。

表 5.1 `ulimit` 参数的含义

参数	含义
-a	显示当前系统所有的 limit 资源信息
-H	设置硬资源限制，一旦设置不能增加
-S	设置软资源限制，设置后可以增加，但是不能超过硬资源设置
-c	最大的 core 文件的大小，以 blocks 为单位
-f	进程可以创建文件的最大值，以 blocks 为单位
-d	进程最大的数据段的大小，以 Kbytes 为单位
-m	最大内存大小，以 Kbytes 为单位
-n	可以打开的最大文件描述符的数量
-s	线程栈大小，以 Kbytes 为单位
-p	管道缓冲区的大小，以 Kbytes 为单位
-u	用户最大可用的进程数

-v	进程最大可用的虚拟内存，以 Kbytes 为单位
-t	最大 CPU 占用时间，以秒为单位
-l	最大可加锁内存大小，以 Kbytes 为单位

在使用 `ulimit` 时，有以下几种使用方法：

(1) 在用户环境变量中加入

如果用户使用的是 `bash`，那么就可以在用户目录的环境变量文件 `.bashrc` 或 `.bash_profile` 中加入 “`ulimit -u 128`” 来限制用户最多可以使用 128 个进程。

(2) 在应用程序的启动脚本中加入

如果应用程序是 `tomcat`，那么就可以在 `tomcat` 的启动脚本 `startup.sh` 中加入 “`ulimit -n 65535`” 来限制用户最多可以使用 65535 个文件描述符。

(3) 直接在 shell 命令终端执行 `ulimit` 命令

这种方法的资源限制仅仅在执行命令的终端生效，在退出或关闭终端后，设置失效，并且这个设置不影响其它 shell 终端。

有时候为了方便起见，也可以将用户资源的限制统一由一个文件来配置，这个文件就是 `/etc/security/limits.conf`，该文件不但能对指定用户的资源进行限制，还能对指定组的资源进行限制。该文件的使用规则如下：

`<domain> <type> <item> <value>`

其中：

- ❑ `domain` 表示用户或组的名字，还可以使用 `*` 作为通配符，表示任何用户或用户组。
- ❑ `type` 表示限制的类型，可以有两个值，`soft` 和 `hard`，分别表示软、硬资源限制。
- ❑ `item` 表示需要限定的资源名称，常用的有 `nofile`、`cpu`、`stack` 等。分别表示最大打开句柄数、占用的 `cpu` 时间、最大的堆栈大小。
- ❑ `value` 表示限制各种资源的具体数值。

除了 `limits.conf` 文件之外，还有一个 `/etc/security/limits.d` 目录，可以将资源限制创建一个文件放到这个目录中，默认系统会首先去读取这个目录下的所有文件，然后才去读取 `limits.conf` 文件。在所有资源限制设置完成后，退出 shell 终端，再次登录 shell 终端后，`ulimit` 设置即可自动生效。

### 3. 解决问题

在介绍了 ulimit 知识后，接着上面的案例，既然 ulimit 设置没问题，那么一定是设置没有生效导致的。接下来检查下启动 tomcat 的 www 用户环境变量下是否添加了 ulimit 限制，检查后发现，www 用户下并无 ulimit 资源限制，于是继续检查 tomcat 启动脚本 startup.sh 文件，是否添加了 ulimit 限制，检查后发现也并无添加，最后考虑是否将限制加到了 limits.conf 文件中，于是检查 limits.conf 文件，操作如下：

```
[root@tomcatserver ~]# cat /etc/security/limits.conf|grep www
www soft nofile 65535
www hard nofile 65535
```

从输出可知，ulimit 限制加在了 limits.conf 文件中，既然限制已经加了，配置也没有错，为何还会报错呢？经过长时间思考，判断只有一种可能，那就是 tomcat 的启动时间早于 ulimit 资源限制的添加时间，于是首先查看下 tomcat 的启动时间，操作如下：

```
[root@tomcatserver ~]# more /etc/issue
CentOS release 6.3 (Final)
Kernel \r on an \m
[root@tomcatserver ~]# uptime
 15:10:19 up 283 days,  5:37,  4 users,  load average: 1.20, 1.41, 1.35
[root@tomcatserver ~]# pgrep -f tomcat
4667
[root@tomcatserver ~]# ps -eo pid,lstart,etime|grep 4667
4667 Sat Jul  6 09:33:39 2013 77-05:26:02
```

从输出看，这台服务器已经有 283 天没有重启过了，而 tomcat 是在 2013 年 7 月 6 日 9 点多启动的，启动了近 77 天零五个半小时了，接着继续看看 limits.conf 文件的修改时间，操作如图 5-19 所示。

```
[root@tomcatserver ~]# stat /etc/security/limits.conf
File: `'/etc/security/limits.conf'
Size: 1880          Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d Inode: 36538264   Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2013-07-12 19:27:10.000000000 +0800
Modify: 2013-07-12 15:48:49.000000000 +0800
Change: 2013-07-12 15:48:49.000000000 +0800
```

图 5-19 查看 limits.conf 文件的最后修改时间

通过 stat 命令可以很清楚地看出，limits.conf 文件最后的修改时间是 2013-07-12，通过查问相关的 Linux 系统管理人员，基本确认就是在这个时候添加的 ulimit 资源限制，这样此案例的问题就很明确了。由于 ulimit 限制的添加时间晚于 tomcat 最后一次的启动时间，而在此期间内，tomcat 服务一直未重启过，操作系统也一直未重启过，那么 ulimit 资源限制对于 tomcat 来说始终是不生效的，同时，由于此操作系统是 Centos6.3，系统默认的最大可用句柄数是 1024，java 进程还是用的 Linux 默认的这个值，因此出现 “Too many open files” 的错误也是合乎情理的。

问题清楚之后，解决问题的方法非常简单，重启 tomcat 服务即可。

## 5.2 Apache 常见错误故障案例

本节主要介绍 Linux 系统下 Web 运维过程中常见的一些故障和案例，这些案例都是基于 Apache 的错误或者故障的，虽然和系统并没有直接关系，但是通过分析和排查发现，其实大部分故障都与系统参数设置有很大关系，所以，下面的案例都基于 Linux 系统的故障进行分析和排查，在介绍过程中，除了给出了故障的解决方法，同时也介绍了与故障相关的 Linux 系统知识，而这些知识是判断和解决问题必不可少的。

### 5.2.1 “No space left on device” 错误与解决方法

#### 1. 错误现象

这也是一个客户案例，客户反映在执行 “apachectl start” 启动 Apache 时无报错信息，但是网页还是不能访问。客户的网站是基于 Apache+PHP+Mysql 的在线交易平台，听到客户描述的现象后，第一反应是防火墙屏蔽了 HTTP 端口或 SELinux 的问题，于是登录服务器查看相关信息，如图 5-20 所示。



```
[root@localhost ~]# more /etc/issue
CentOS release 5.5 (Final)
Kernel \r on an \m
[root@localhost ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination |
[root@localhost ~]# more /etc/selinux/config |grep -v "#"|grep SELINUX
SELINUX=disabled
SELINUXTYPE=targeted
```

图 5-20 查看系统防火墙和 SELinux 状态

从输出可知，防火墙所有策略都处于开放状态，并未作任何限制，而 SELinux 也处于关闭状态，应该不是防火墙问题导致的。

既然不是防火墙拦截的问题，那么看看 httpd 进程是否存在及 httpd 端口是否正常启动，操作过程如图 5-21 所示。

```
[root@localhost ~]# ps -ef|grep httpd|grep -v "grep"|wc -l
0
[root@localhost ~]# netstat -antl|grep 80
[root@localhost ~]# /usr/local/apache2/bin/apachectl start
[root@localhost ~]# ps -ef|grep httpd|grep -v "grep"|wc -l
0
```

图 5-21 查看 httpd 进程运行状态

这个操作首先查看了服务器上的 httpd 进程，发现并没有 httpd 进程运行，同时 httpd 对应的端口 80 也并无启动，于是重新启动 Apache，在启动 Apache 的过程中并没有报错，启动完成后发现仍然没有 httpd 进程运行，由此看来，应该是 Apache 内部出现了问题。

## 2. 解决思路

在判定是 Apache 的问题后，首先要看的就是 Apache 的启动日志，在查看 Apache 的 error 日志后，发现了一个可疑输出，内容为：

```
No space left on device: mod_rewrite: could not create rewrite_log_lock Configuration Failed
```

看到这个错误提示，感觉应该是磁盘空间耗尽导致的，于是赶紧查看系统所有磁盘分区，结果发现所有磁盘分区都还有很多可用空间，这就有些奇怪了。

在前面的案例介绍中，详细介绍了 Linux 下对磁盘空间的占用分为三个部分：物理磁盘、

inode 节点磁盘空间和信号量磁盘空间。通过检查服务器的物理磁盘空间，发现仍有很多剩余，因此就被排除了物理磁盘空间的问题。接着通过“df -i”命令查看系统可用的 inode 节点，发现每个分区可用的 inode 还有很多，这样 inode 节点问题也被排除了，那么应该是信号量磁盘空间耗尽导致的。

这里简单介绍下 Linux 信号量相关的知识，信号量是一种锁机制用于协调进程之间互斥的访问临界资源，以确保某种共享资源不被多个进程同时访问。Linux 系统的信号量是用于进程间通信的。它有两种标准实现，分别是 POSIX 及 System v，现在大多数 Linux 系统都实现了这两种标准。这两种标准都可用于进行线程间的通信，只是系统调用方式略有不同。

- ❑ System v 信号量通过系统调用 semget 来创建，通过 Linux 命令 ipcs 即可显示进程间通信用的 System v 类型信号量及共享内存。
- ❑ POSIX 信号量可用于线程和进程间通信，并可分为有名和无名两种。也可简单理解为是否保存在磁盘上。有名的信号量会以文件形式保存在/dev/shm 下，因此可用于不相关的进程间通信，而无名信号量只能用于线程间和父子进程间通信。

在对信号量有了简单了解后，可以发现 Apache 使用的进程间通信方式应该是 System v，因此就可以通过 ipcs 命令查看和解决这个问题了。

### 3. 解决问题

在解决这个问题之前，首先查看下 Linux 系统默认信号量的设置值是多少，执行如下命令：

```
[root@localhost ~]#cat /proc/sys/kernel/sem
250 32000 32 128
```

这 4 个输出值的含义如下：

- ❑ SEMMSL，此参数用于控制每个信号集的最大信号数。
- ❑ SEMMNS，此参数用于控制整个 Linux 系统中信号（而不是信号集）的最大数量。
- ❑ SEMOPM，此参数用于控制每个 semop 系统调用可以执行的信号操作数。
- ❑ SEMMNI，此内核参数用于控制整个 Linux 系统中信号集的最大数量。

接着通过 ipcs 命令查看 httpd 进程占用了多少信号量，执行如下命令：

```
[root@localhost ~]#ipcs -s | grep daemon
```

其中“daemon”是启动 Apache 进程的用户，默认是 daemon，也可能是 nobody 用户，

根据实际环境而定。在执行完此命令后，发现很多基于 `daemon` 的信号量输出，问题终于找到了。解决信号量耗尽的方法很简单，通过 `ipcrm` 命令清除即可，最简单方法是执行如下命令组合：

```
[root@localhost ~]#ipcs -s | grep nobody | perl -e 'while (<STDIN>) { @a=split(/\s+/); print `ipcrm sem $a[1]`}'
```

执行完上面这个命令后，再次启动 Apache，然后查看是否有 `httpd` 进程启动，可以看到，此时 `httpd` 进程启动了，那么 Apache 也工作正常了。

## 5.2.2 apache(20014)故障与解决方法

### 1. 错误现象

这是一个客户的简单案例。客户告知网站无法访问了，并且 Apache 也无法启动。客户的网站是基于 Apache+Tomcat+Mysql 构建，于是登录服务器查看信息如图 5-22 所示。

```
Linux:~ # uname -a
Linux linux 2.6.5-7.97-default #1 Fri Jul 2 14:21:59 UTC 2004 i686 i686 i386 GNU/Linux
linux:~ # more /etc/issue
Welcome to SUSE LINUX Enterprise Server 9 (i586) - Kernel \r (\l).
linux:~ # /usr/local/apache2/bin/apachectl start
(20014)Internal error: Error retrieving pid file logs/httpd.pid
```

图 5-22 Apache (20014)Internal error 现象

这里提示的是 `httpd.pid` 文件的错误，熟悉 Apache 的读者应该知道 `httpd.pid` 文件其实是 Apache 的进程 `pid` 文件，Apache 的启动进程 ID 就放在这个文件中。

### 2. 解决思路

既然提示 `httpd.pid` 这个文件有问题，那么就先看看这个文件是否存在，以及其中的内容是什么。查看 `httpd.pid` 文件，操作如下：

```
linux:~ #more /usr/local/apache2/logs/httpd.pid
```

发现这个文件存在，但是内容为空，这里肯定是有问题的。要解决这个问题，首先要了解 Apache 的启动机制及 `httpd.pid` 文件的作用。

`httpd.pid` 文件为文本文件，内容只有一行，记录了 `httpd` 进程的 `pid`。通过 `cat` 命令可以看到该文件的内容，通过这个 `pid` 文件可以防止进程启动多个副本。只有能获得 `pid` 文件的

进程才能正常启动并把自身的 pid 写入该文件中。同一个程序的多余进程则自动退出。同时，httpd.pid 文件在 Apache 正常启动时创建，正常关闭时自动删除，Apache 在启动时会查找 httpd.pid 文件是否存在，如果文件不存在就创建此文件，将 Apache 启动的进程 ID 写入 httpd.pid 中，并提示启动成功。如果文件存在但内容为空，那么就会出现“(20014)Internal error”的错误了。

在这个案例中，httpd.pid 文件存在，但是内容为空，这个现象可能是磁盘空间耗尽导致的，也可能是系统突然断电导致的，总之导致这个问题的原因有很多种，这里并不打算深究，因为找到解决问题的办法是我们的目的。

### 3. 解决问题

解决这个问题有两个办法：可以直接删除 httpd.pid 这个空文件，也可以将这个文件写入一个数字 ID 值，操作如下：

```
linux:~ #echo "28976">>/usr/local/apache2/logs/httpd.pid
```

```
linux:~ #more /usr/local/apache2/logs/httpd.pid
```

```
28976
```

接着，再次启动 apache：

```
linux:~ #/usr/local/apache2/bin/apachectl start
```

这次 Apache 可以正常启动了，此时查看 httpd.pid 文件内容，信息如下：

```
linux:~ #more /usr/local/apache2/logs/httpd.pid
```

```
7789
```

可以看到，Apache 成功启动后，已经自动获得了一个新的 pid 值，进程间通信就以这个 pid 进行。Apache 在启动后，保持这个 pid 值不变，直到下次重新启动 Apache 才更新 httpd.pid 的值。

## 5.2.3 “could not bind to address 0.0.0.0:80” 错误与解决方法

### 1. 错误现象

客户的一台 Web 服务器，是基于 Apache+JK+Tomcat 构建的一个电商平台，在机器更换硬件重新启动后，客户反映 Apache 无法启动，但是 tomcat 可以启动，启动 Apache 的错

机械工业出版社出版发行

误信息如图 5-23 所示。

```
[www@cloud1 ~]$ /usr/local/apache2/bin/apachectl start
(13)Permission denied: make_sock: could not bind to address [::]:80
(13)Permission denied: make_sock: could not bind to address 0.0.0.0:80
no listening sockets available, shutting down
Unable to open logs
```

图 5-23 Apache 无法启动错误现象

于是又检查了操作用户和 Apache 监听的端口，如图 5-24 所示。

```
[www@cloud1 apache2]$ whoami
www
[www@cloud1 apache2]$ cat /usr/local/apache2/conf/httpd.conf |grep Listen
# Listen: Allows you to bind Apache to specific IP addresses and/or
# Change this to Listen on specific IP addresses as shown below to
#Listen 12.34.56.78:80
Listen 80
```

图 5-24 查看 Apache 的管理用户和监听端口

从输出可知，Apache 的启动用户是 www，监听端口为 80，接着查看到/usr/local/apache2 目录所有文件和目录的权限都是 www，看来不是读写权限的问题，于是继续排查，这里更换了 Apache 的监听端口，将其改为 8000，看是否能启动成功，操作如图 5-25 所示。

```
[www@cloud1 apache2]$ sed -i 's/Listen 80/Listen 8000/' /usr/local/apache2/conf/httpd.conf
[www@cloud1 apache2]$ cat /usr/local/apache2/conf/httpd.conf |grep Listen
# Listen: Allows you to bind Apache to specific IP addresses and/or
# Change this to Listen on specific IP addresses as shown below to
#Listen 12.34.56.78:80
Listen 8000
[www@cloud1 apache2]$ /usr/local/apache2/bin/apachectl start
[www@cloud1 apache2]$ ps -ef|grep httpd
www      31756      1   0 18:39 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
www      31758 30870   0 18:39 pts/4    00:00:00 grep httpd
```

图 5-25 修改 Apache 监听端口为 8000 后重启 Apache 服务

虽然这次启动没报错，但是根据 httpd 进程的状态来看，启动应该还有问题，于是查看 Apache 启动日志，如图 5-26 所示。

```
[Mon Sep 16 18:39:40 2013] [warn] pid file /usr/local/apache2/logs/httpd.pid overwritten -- Unclean shutdown of previous Apache run?
[Mon Sep 16 18:39:40 2013] [error] (13)Permission denied: could not create /usr/local/apache2/logs/httpd.pid
[Mon Sep 16 18:39:40 2013] [error] httpd: could not log pid to file /usr/local/apache2/logs/httpd.pid
```

图 5-26 Apache 启动错误信息

从日志输出看，果然存在问题，通过日志基本判断是 Apache 的 pid 文件无权限导致的。接着检查 httpd.pid 文件的权限，操作如下：

```
[root@cloud1 logs]# ll /usr/local/apache2/logs/httpd.pid
-rw-r--r-- 1 root www          6 Sep 16 17:33 /usr/local/apache2/logs/httpd.pid
```

从输出可知，httpd.pid 权限的属主为 root，将其修改为 www 用户，操作如下：

机械工业出版社出版发行

```
[root@cloud1 logs]# chown www /usr/local/apache2/logs/httpd.pid
```

然后，重启 Apache，操作如图 5-27 所示。

```
[www@cloud1 apache2]$ /usr/local/apache2/bin/apachectl restart
httpd not running, trying to start
[www@cloud1 apache2]$ ps -ef|grep httpd
www      31756      1  0 18:39 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
www      32408      1  0 18:43 ?        00:00:00 /usr/local/apache2/bin/httpd -k restart
www      32409 32408    0 18:43 ?        00:00:00 /usr/local/apache2/bin/httpd -k restart
www      32410 32408    0 18:43 ?        00:00:00 /usr/local/apache2/bin/httpd -k restart
www      32411 32408    0 18:43 ?        00:00:00 /usr/local/apache2/bin/httpd -k restart
www      32414 32408    0 18:43 ?        00:00:00 /usr/local/apache2/bin/httpd -k restart
www      32573 30870    0 18:43 pts/4    00:00:00 grep httpd
```

图 5-27 重启 Apache 服务并查看启动进程

可以看到，这次 Apache 启动成功了，看来 Apache 配置并无问题，通过 8000 端口可以启动，而通过 80 端口则无法启动，这是什么问题呢？

## 2. 解决思路

既然这个案例是与端口相关的，那么就需要了解下 Linux 系统中的端口，在 Linux 系统下可用的端口范围是 1~65535，端口可分为三类，分别是公认端口、注册端口和动态端口。

- ❑ 公认端口的范围为 0~1023，属于系统预留端口，主要用于绑定一些服务，例如 80 端口绑定的是 http 服务，21 端口绑定的是 ftp 服务，22 端口绑定的是 sshd 服务等。对于公认端口，Linux 系统做了一些安全限制，那就是普通用户无法绑定这类端口，只有 root 用户才能绑定和使用公认端口。
- ❑ 注册端口的范围是 1024~49151，主要用于分配给用户进程或应用程序，这些进程主要是用户自定义安装的一些应用程序。
- ❑ 动态端口的范围是从 49152~65535，动态分配是指当一个系统进程或应用程序需要进行网络通信时，它就向主机申请一个端口，主机从可用的端口号中分配一个供它使用。当这个进程关闭时，同时也就释放了所占用的这个端口。

通过以上对端口的介绍，这个案例描述的问题也就明确了，在上面的错误现象中，普通用户 www 无法启动 Apache 的 80 端口，就是因为 80 端口属于公认端口，普通用户无权绑定，而 8000 端口属于注册端口，普通用户可以自由使用，这就是此案例要查找的原因。

## 3. 解决问题

如何使用 Apache 的 80 端口呢，这里提供两种方法，分别是：



- 1) 将 Apache 以 root 用户启动即可，这是最简单的方法。
- 2) 修改 Apache 目录下 httpd 文件的 SUID 属性。

第一种方法实现简单，但是有安全问题，如果黑客入侵了 80 端口，那么他也就拥有了 root 权限，因此，不推荐使用第一种方法，因为保证程序的安全才是根本。这里简单介绍下第二种方法的实现过程。首先通过 root 用户进行如下授权，如图 5-28 所示。

```
[www@cloud1 apache2]#chmod u+s /usr/local/apache2/bin/httpd
[www@cloud1 apache2]#chown root /usr/local/apache2/bin/httpd
```

图 5-28 对 Apache 的 httpd 文件进行授权

然后在 www 用户下再次启动 Apache，可以看到这次启动正常了，如图 5-29 所示。

```
[www@cloud1 apache2]$ ps -ef|grep httpd
root      12892      1   0 19:46 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
www       12893 12892   0 19:46 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
www       12894 12892   0 19:46 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
www       12895 12892   1 19:46 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
www       12896 12892   0 19:46 ?        00:00:00 /usr/local/apache2/bin/httpd -k start
www       12983 30870   0 19:46 pts/4    00:00:00 grep httpd
```

图 5-29 通过 www 用户重启来启动 Apache 服务

从这个输出可以看出，其实 Apache 还是在 root 用户下启动的，上面的修改只不过是保证普通用户可以正常启动 Apache 而已，另外，启动的 httpd 进程对应的用户除了 root，还有 www 用户，这其实是一种父进程和子进程的关系，父进程由 root 用户启动后，会派生出多个子进程，而这些子进程的启动用户是可定义的，可以在配置文件 httpd.conf 的如下选项中修改：

User www

Group www

至此，这个案例被完整剖析！

## 5.3 因 NAS 存储故障引起的 Linux 系统恢复案例

### 5.3.1 故障现象描述

NAS 操作系统内核为 Linux，自带的存储有 16 块硬盘，总共分两组，每组都做了 RAID5，Linux 操作系统无法正常启动，在服务启动到 cups 那里就停止了，按 ctrl+c 强制断开也没有响应，查看硬盘状态，都是正常的，没有报警或警告现象。

## 5.3.2 问题判断思路

通过上面这些现象，首先判断 NAS 硬件应该没问题，NAS 存储盘也应该正常，现在 Linux 无法启动，应该是 Linux 系统本身存在问题，因此，首先从 Linux 系统入手进行排查。

## 5.3.3 问题处理过程

### 1. 第一次处理过程

NAS 系统本身就是一个 Linux 内核装载了一个文件系统管理软件，管理软件可以对系统磁盘、系统服务、文件系统等进行管理和操作，在正常情况下，基于 Linux 内核的 NAS 系统应该启动到 `init3` 或 `init5` 模式下，由于 NAS 仅用了 Linux 一个内核模块和几个简单服务，因此判断 NAS 下的 Linux 系统肯定是启动到 `init 3` 模式下，那么既然现在无法启动到多用户字符界面下，何不让 Linux 直接进入单用户 (`init 1`) 模式下呢？因为单用户模式下仅仅启用系统所必须的几个服务，而 `cpus` 服务是应用程序级别的，肯定会在“`init 1`”模式下启动，这样就避开了 `cpus` 无法启动的问题，所以，下面的工作就是要进入 Linux 的单用户模式下。

很多的 Linux 发行版本都可以在启动的引导界面通过相关的设置进入单用户模式下，通过查看 NAS 的启动过程，基本判断这个 Linux 系统与 RHEL/Centos 发行版极为类似，因此，就通过 RHEL/Centos 进入单用户模式的下试一试。

RHEL/Centos 进入单用户模式很简单，就是在系统启动到引导欢迎界面下时，按 `e` 键，然后编辑正确的内核引导选项，在最后面加上“`single`”选项，最后直接按 `b` 键即可进入单用户了。

接下来，重新启动 NAS，然后硬件自检，接着开始启动 Linux，一直在等待这个 NAS 的启动欢迎界面，但是欢迎界面一直没出来，就直接进入内核镜像开始加载内核阶段了。没有内核引导界面，如何进入单用户呢？经过简单思考，还是决定在硬件检测完毕后直接按 `e` 键，奇迹出现了，真的可以，NAS 进入到了内核引导界面！通过简单观察，发现第二个正是要引导的内核选项，于是移动键盘中的上下键，选择这个内核，然后按 `e` 键，进入内核引导编辑界面了，在这行的最后面输入“`single`”，然后按回车键，返回上个界面，接着按 `b` 键开始进行单用户引导，经过一分钟的时间，系统如愿以偿地进入了单用户下的 `shell` 命令行。

进入单用户模式后，能做的事情就很多了，首先要做的就是将 cups 服务在多用户模式下自启动关闭，执行命令如下：

```
chkconfig --level 35 cups off
```

执行成功后，重启系统进入多用户模式下，看看系统是否能正常启动。

## 2. 第二次处理过程

将 cups 服务开机自启动关闭后，重启 NAS，发现问题依旧，NAS 还是在启动到 cups 服务那里停止了，难道上面的命令没有执行成功吗？明明已经禁止了 cups 服务启动了，怎么还是启动了呢？于是，继续重启 NAS，再次进入单用户模式下，看看问题究竟出在哪里了。

进入单用户后，再次执行 chkconfig 命令，依旧可以成功，难道是 cups 服务有问题，先看看配置文件，执行如下命令：

```
vi /etc/cups/cupsd.conf
```

在这里发现了一个问题，在通过 vi 命令打开 cupsd.conf 时，提示“write file in swap”，文件明明真实存在，怎么说在虚拟内存中呢？经过思考，只有一种可能，NAS 设备的 Linux 系统分区应该没有正确挂载，导致在进入单用户的时候，所有文件都存储在了虚拟内存中，要验证这一点非常简单，执行“df”命令查看即可，如图 5-30 所示。

```
[root@NASserver ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/           11G   4.3G   5.8G  43% /
tmpfs           593M   4.0K   593M   1% /dev/shm
```

图 5-30 查看 NAS 挂载磁盘分区情况

从这里可以看出，Linux 的系统分区并未挂载，通过“fdisk -l”检查磁盘分区状态，输出如图 5-31 所示。

```
[root@NASserver ~]# fdisk -l

Disk /dev/sda: 85.8 GB, 85899345920 bytes
255 heads, 63 sectors/track, 10443 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *           1           13        104391   83  Linux
/dev/sda2                14        10443       83778975   8e  Linux

Disk /dev/sdb: 1999.8 GB, 1999844147200 bytes
255 heads, 63 sectors/track, 243133 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1  *           1        243133    1952965791   83  Linux

Disk /dev/sdc: 1999.8 GB, 1999844147200 bytes
255 heads, 63 sectors/track, 243133 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1  *           1        243133    1952965791   83  Linux
```

图 5-31 查看 NAS 磁盘分区情况

通过输出可知,NAS 的系统盘是/dev/sda,仅划分了/dev/sda1 和/dev/sda2 两个系统分区,而数据磁盘是经过做 RAID5 完成的,在系统上的设备标识分别是/dev/sdb1 和/dev/sdc1。由于单用户默认没有挂载任何 NAS 磁盘,这里尝试手动挂载 NAS 的系统盘,执行如下命令:

```
[root@NASserver ~]#mount /dev/sda2 /mnt
```

```
[root@NASserver ~]#mount /dev/sda1 /opt
```

这里的/mnt、/opt 是随意挂载的目录,也可以挂载到其他空目录下,完成挂载后分别进入这两个目录看看内容有什么,如图 5-32 和 5-33 所示。

```
[root@NASserver opt]# cd /mnt
[root@NASserver mnt]# ls -al
drwxr-xr-x  2 root  root   4096 Apr 12 14:23 bin
drwxr-xr-x 11 root  root   4420 Jul  6 14:37 dev
drwxr-xr-x  5 root  root   4096 Jul 18 15:11 home
drwxr-xr-x 97 root  root  12288 Sep 18 04:26 etc
drwxrwxrwt  8 root  root   4096 Sep 21 04:02 tmp
drwxr-xr-x 18 root  root   4096 Jul 18 21:27 usr
drwxr-xr-x 22 root  root   4096 Apr 12 13:09 var
```

图 5-32 查看/dev/sda2 磁盘标识下的内容

```
[root@NASserver ~]# cd /opt
[root@NASserver opt]# ls -al
total 6226
drwxr-xr-x  4 root root   1024 Apr 12 13:10 .
drwxr-xr-x 36 root root   4096 Jul  6 14:36 ..
-rw-r--r--  1 root root  67541 Feb 22  2012 config-2.6.18-308.el5
drwxr-xr-x  2 root root   1024 Apr 12 13:11 grub
-rw-----  1 root root 2651208 Apr 12 13:10 initrd-2.6.18-308.el5.img
drwx-----  2 root root   12288 Apr 12 13:06 lost+found
-rw-r--r--  1 root root   80032 Mar 13  2009 message
-rw-r--r--  1 root root  116635 Feb 22  2012 symvers-2.6.18-308.el5.gz
-rw-r--r--  1 root root 1275921 Feb 22  2012 System.map-2.6.18-308.el5
-rw-r--r--  1 root root 2115772 Feb 22  2012 vmlinuz-2.6.18-308.el5
```

图 5-33 查看/dev/sda1 磁盘标识下的内容

通过这两个内容的查看，初步判断，/dev/sda2 分区应该是 Linux 的根分区，而/dev/sda1 应该是/boot 分区。现在分区已经挂载上去了，再次执行 df 命令看看挂载情况，如图 5-34 所示。

```
[root@NASserver ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/           11G   4.3G  5.8G  43% /
tmpfs           8G     0   8G   0% /dev/shm
/dev/sda2       72G   72G   0  100% /mnt
/dev/sda1       99M   20M  75M  21% /opt
```

图 5-34 查看 NAS 系统盘的磁盘空间

到这里为止，发现问题了。/dev/sda2 磁盘分区已经没有可用的磁盘空间了，而这个分区刚好是 NAS 系统的根分区，根分区没有空间了，那么系统启动肯定就出问题了。

下面再把思路转到前面介绍的案例中，由于系统 cups 服务在启动的时候会写启动日志到根分区，而根分区没有空间了，因此也就无法写日志了，由此导致的结果就是 cups 服务无法启动，这就解释了此案例中 NAS 系统每次启动到 cups 服务就停止的原因。

### 5.3.4 解决问题

由于 NAS 系统只有根分区和/boot 分区，因此系统产生的相关日志都会存储在根分区中，现在根分区满了，首先可以清理的就是/var 目录下的系统相关日志文件，通常可以清理的目录有/var/log。执行如下命令查看/var/log 日志目录占据磁盘空间大小：

```
[root@NASserver ~]# du -sh /var/log
50.1G    /var/log
```

通过命令输出发现/var/log 目录占据了根分区仅 70%的空间，清理这个目录下的日志文

件即可释放大部分根分区空间，清理完毕后重启 NAS 系统，发现系统 cups 服务能正常启动了，NAS 服务也启动正常了。

高性能Linux服务器构建实战II