目录

[1、 文件系统破坏导致系统无法启动 1](#_Toc517197315)

[2、 “Argument list too long”错误与解决方法 1](#_Toc517197316)

[3、 Inode被耗尽导致应用程序无法启动 2](#_Toc517197317)

[4、 文件已经删除，但是空间没有释放的原因 3](#_Toc517197318)

[5、 "too many open files" 错误与解决方法 4](#_Toc517197319)

[6、 Read-only file system 错误与解决方法 5](#_Toc517197320)

## 

## 文件系统破坏导致系统无法启动

**问题描述：**

启动系统提示如下错误

Checking root filesystem

/dev/sda6 **contains** a file system **with** errors, **check** forced

An error occurred during the file system **check**

通过报错可以看出是/dev/sda6文件系统出现了故障，此问题可能是由于系统断电导致。

**解决方法：**

解决此问题可以通过fsck命令进行修改；

fsck.ext3 –y /dev/sda6

## “Argument list too long”错误与解决方法

**问题描述：**

当我们在执行一个命令时，报“no space left on device”

根据报错信息我们了解到是磁盘空间满了，那么我们通过检查磁盘空间（df -h）发现一个分区满了；

通过du –sh\* 查看目录下最哪些目录使用的空间最大，例如：

[root@mysqldb var]# du -sh \*

0 adm

1.5M cache

0 crash

8.0K db

当发现一个目录的空间很大时，在确认该目录的文件可以删除后使用rm –rf \*时提示如下：

# rm \*

/bin/rm :argument list too long

此问题是由于Linux系统限制，无法为命令添加太多的参数，可通过命令查看；

[root@mysqldb var]# getconf ARG\_MAX

2097152

**解决方法：**

1、分批删除

rm [a-n]\* -rf

rm [o-z]\* -rf

2、使用 find 命令来删除

find /var/spool/clientmqueue –type f –print **–exec rm –f {}**

3、通过 shell 脚本

#/bin/bash

RM\_DIR=’/var/spool/clientmqueue’

cd $RM\_DIR

for I in `ls`

do

rm –f $i

done

4、重新编译内核

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：

denfine MAX\_ARG\_PAGES 32

将 32 改为更大的值，例如 64 或者 128，然后重新编译内核

## Inode被耗尽导致应用程序无法启动

**问题描述：**

客户的一台 Oracle 数据库如武器在关机重启后，Oracle 监听无法启动，提示报错Linux error : No space left on device

从输出信息看出来是因为磁盘耗尽导致监听无法启动，因为 Oracle 在启动监听时需要创建监听日志文件，于是首先查看磁盘空间使用情况

df -h

从磁盘输出信息可知，所有的分区磁盘空间都还有剩余不少，而 Oracle 监听写日志的路径在 / var 分区下，/var 下分区空间足够。

解决思路：

既然错误提示语磁盘空间有关，那就深入研究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。既然不是物理磁盘空间的问题，接着就检查是否是 inode 节点耗尽的问题，通过执行命令 “df -i” 查看可用的 inode 节点。由输出结果看出确实是因为 inode 耗尽导致无法写入文件。

[root@mysqldb var]# df -i

Filesystem Inodes IUsed IFree IUse% Mounted on

/dev/mapper/cl-root 8910848 55377 8855471 1% /

devtmpfs 230662 392 230270 1% /dev

tmpfs 233412 1 233411 1% /dev/shm

tmpfs 233412 488 232924 1% /run

tmpfs 233412 16 233396 1% /sys/fs/cgroup

/dev/sda1 524288 330 523958 1% /boot

tmpfs 233412 1 233411 1% /run/user/0

可以通过下面的命令查看某个磁盘分区 inode 的总数

dumpe2fs -h /dev/sda3 |grep ‘Inode count’

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 -i’命令可以查看文件名对应的 inode 号

如果要查看这个文件更详细的 inode 信息，可以通过 stat 命令来实现

# stat install.log

**解决方法：**

find /var/spool/clientmqueue/ -name “\*” -exec rm -rf {} ;

## 文件已经删除，但是空间没有释放的原因

运维监控系统发来通知，报告一台服务器空间满了，登陆服务器查看，根分区确实满了，这里先说一下服务器的一些删除策略，由于 linux 没有回收站功能，所以线上服务器上所有要删除的文件都会先移到系统 / tmp 目录下，然后定期清除 / tmp 目录下的数据。这个策略本身没有什么问题，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 下的数据其实占用根分区的空间，既然找到了问题，那么删除 / tmp 目录下一些占用空间较大的数据文件即可。

du -sh /tmp/\* | sort -nr |head -3

通过命令发现在 / tmp 目录下有个 66G 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的 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，执行如下删除命令，

rm /tmp/access\_Iog

df -h

从输出来看，根分区空间仍然没有释放，这是怎么回事

一般来说不会出现删除文件后空间不释放的情况，但是也存在例外，比如文件进程锁定，或者有进程一直在向这个文件写数据，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，之所以出现删除 access\_log 文件后，空间还没有释放，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\_Ilog 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。

**问题排查：**

既然有了解决思路，那么接下来看看是否有进程一直在向 access\_log 文件中写入数据，这里需要用到 linux 下的 losf 命令，通过这个命令可以获取一个仍然被应用程序占用的已删除文件列表（lsof命令需要安装lsof软件）

lsof | grep delete

从输出可以看出，/tmp/access\_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，最后一列的‘deleted’状态说明这个日志文件已经被删除，但是由于进程还在一直向此文件写入数据，因此空间并未释放。

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，最简单的方法就是关闭或者重启 httpd 进程，当然重启操作系统也可以。不过这些并不是最好的办法，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:

echo “”>/tmp/access\_log

通过这种方法，磁盘空间不但可以马上释放，也可以保障进城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。

## "too many open files" 错误与解决方法

**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，于是登陆服务器查看 tomcat 日志，发现如下异常信息，java.io.IOException: Too many open files

通过这个报错信息，基本判断是系统可以用的文件描述符不够了，由于 tomcat 服务室系统 www 用户启动的，于是以 www 用户登陆系统，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：

[root@mysqldb var]# ulimit -a

core file size (blocks, -c) 0

data seg size (kbytes, -d) unlimited

scheduling priority (-e) 0

file size (blocks, -f) unlimited

pending signals (-i) 7208

max locked memory (kbytes, -l) 64

max memory size (kbytes, -m) unlimited

open files (-n) 65535

pipe size (512 bytes, -p) 8

POSIX message queues (bytes, -q) 819200

real-time priority (-r) 0

stack size (kbytes, -s) 8192

cpu time (seconds, -t) unlimited

max user processes (-u) 7208

virtual memory (kbytes, -v) unlimited

file locks (-x) unlimited

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

解决思路，这个案例涉及 ulimit 命令的使用

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，那么可以在用户目录的环境变量文件. bashrc 或者. bash\_profile 中加入 “ulimit –u128” 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，那么可以再 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，在退出或者和关闭终端后，设置失效，并且这个设置不影响其他 shell 终端

解决问题：

在了解 ulimit 知识后，接着上面的案例，既然 ulimit 设置没有问题，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 用户环境变量是否添加 ulimit 限制，检查后发现，www 用户并无 ulimit 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，检查后发现也没有添加。最后考略是否将限制加到了 limits.conf 文件中，于是检查 limits.conf 文件，操作如下

# 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 limits.conf 文件中，既然限制已经添加了，配置也没有什么错，为何还会报错，经过思考，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 启动时间，操作如下

# uptime

Up 283 days

# pgrep -f tomcat

4667

# ps -eo pid,lstart,etime|grep 4667

4667 Sat Jul 6 09;33:39 2013 77-05:26:02

从输出可以看出，这台服务器已经有 283 没有重启了，而 tomcat 是在 2013 年 7 月 6 日 9 点启动的，启动了将近 77 天，接着继续看看 limits.conf 文件的修改时间，

# stat /etc/security/limits.conf

通过 stat 命令清除的看到，limits.conf 文件最后的修改时间是 2013 年 7 月 12，晚于 tomcat 启动时间，清楚问题后，解决问题的方法很简单，重启一下 tomcat 就可以了。

## Read-only file system 错误与解决方法

解析：出现这个问题的原因有很多种，可能是文件系统数据块出现不一致导致的，也可能是磁盘故障造成的，主流 ext3/ext4 文件系统都有很强的自我修复机制，对于简单的错误，文件系统一般都可以自行修复，当遇到致命错误无法修复的时候，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，讲文件系统 变为只读，今儿出现了上面的 “read-only file system” 现象。

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

# umount /www/data

Umount : /www/data: device is busy

提示无法卸载，可能是这个磁盘中还有文件对应的进程在运行，检查如下：

# fuser -m /dev/sdb1

/dev/sdb1: 8800

接着检查一下 8800 端口对应的什么进程，

# ps -ef |grep 8800

检查后发现时 apache 没有关闭，停止 apache

# /usr/local/apache2/bin/apachectl stop

# umount /www/data

# fsck -V -a /dev/sdb1

# mount /dev/sdb1 /www/data