

刪庫了

原創 小林coding 小林coding 昨天

大家好，我是小林。

在知乎看到這麼個問題「**不小心刪庫是一種怎樣的體驗？**」

数据库

程序员

实习生

计算机科学

不小心删库是一种怎样的体验？

已关注

[查看回答](#)[邀请回答](#)[好问题 4](#)[23 条评论](#)[分享](#)

...

[查看全部 618 个回答](#)**小林coding**

公众号：「小林coding」 专注图解计算机基础

四猿外、一只自动编码器、码农的荒岛求生、张彦飞、程序喵大人等 31 人赞同了该回答

这事我干过，我不过我不是删数据库这么简单，而是直接 `rm -fr /*`，直接把公司的服务器删了，当时是写了个 Bug，导致误执行了这条命令，吓的一身冷汗！



不知道大家幹過刪庫的事情嗎？

還別說，這事小林還真幹過，不過我不是刪數據庫這麼簡單，而是直接 `rm -fr /*`，更糟心的是在公司的服務器做的。

別問為什麼我有權限執行，因為這個台服務器歸我管的，我都是直接root 權限登錄的。

當時是寫了個Bug，導致誤執行了這條命令，**我這年紀輕輕的身子瞬間被嚇的一身冷汗！**

不過，這件事是去年的事情，然後最近看到群裡有小伙伴也誤執行了刪庫命令，翻了我刪庫的文章，找到了恢復的思路。

雖然這篇文章去年發過，但是當時關注的人少看的人不多，所以今天再分享給大家。

我來跟大家復盤下，當時的事件經過，挺有意思的。

事件開始

去年臨近五一節，想到有5天假期，我就開始飄了。寫個簡單的 **Bash** 腳本都不上心了，寫完連檢查都不檢查，直接拖到到實體服務器跑。

結果一跑起來，發生不對勁，怎麼一個簡單腳本跑了10秒還沒結束，於是**立馬**直接 **ctrl + c** 一頓操作停掉了運行中腳本。

接著，習慣性的輸入了 **ls**，結果what？找不到 **ls** 命令？

```
[root@servermanager /]# ls
-bash: /bin/ls: No such file or directory  ? ? ?
[root@servermanager /]# ls
-bash: /bin/ls: No such file or directory
[root@servermanager /]# ls
-bash: /bin/ls: No such file or directory
[root@servermanager /]# ls
-bash: /bin/ls: No such file or directory
```

瞬間背後一涼，慌慌張張打開了腳本。

發現問題了，小林我寫了個巨蠢的Bug，間接執行了 `rm -fr /*`，這不意味著我刪庫了？



笑容漸漸呆滯

這台是公司內部使用的授權服務器呀，被小林這麼一整，公司歷史的授權記錄和其他重要信息不就丟了？

心裡慌的一批的小林，跟我的朋友們說了這件事，朋友建議我先第一時間上報給leader，不要把刪庫的事情瞞著。

於是，小林就向leader 說了我刪庫事情，本以為會被痛批一頓。

結果leader**笑著**說：“沒事，你先看看重要的文件還在不在。不過你這麼一整，我突然想起編譯服務器半年沒備份，我先備份一下我的編譯服務器，防止哪天也被你們刪庫了。”

我：“????”

吃瓜的小伙伴，是不是覺得小林要刪庫跑路了？

哈哈，小林沒跑路，反而是恢復了回來，所以接下來說說小林是如何「**從刪庫到恢復**」的。

初探案發現場

來看看小林寫的垃圾代碼，是如何引發這次的刪庫。

```
lic_path=`pwd`  
new_lic_dir=`${lic_path}/new_license`  
old_lic_dir=`${lic_path}/old_license`  
  
rm -fr $new_lic_dir/* 2>>/dev/null  
rm -fr $old_lic_dir/* 2>>/dev/null
```

既然發生了 `rm -fr /*` 的現象，那必然 `new_lic_dir` 這個變量是空的。

所以導致執行 `rm -fr $new_lic_dir/*` 這條語句的時候，變成了 `rm -fr /*` 刪庫語句。很好，凶器找到了。

那為什麼 `new_lic_dir` 會是空的呢？

細心的小伙伴肯定察覺出來了，是因為給 `new_lic_dir` 變量賦值的時使用了反引號。

```
lic_path=`pwd`  
new_lic_dir=`${lic_path}/new_license`  
old_lic_dir=`${lic_path}/old_license`  
  
rm -fr $new_lic_dir/* 2>>/dev/null  
rm -fr $old_lic_dir/* 2>>/dev/null
```

沒錯，就是反引號的原因。

反引号在 Linux Shell 命令行中有特殊的含义：反引号间的内容，会被 Shell 先执行。其输出被放入主命令后，主命令再被执行。

也就是说，`new_lic_dir` 的值是 `${lic_path}/new_license` 这条命令执行的结果，问题这哪是命令啊，所以肯定返回空值给 `new_lic_dir` 变量。

小林写的那么温柔的代码，竟然变成了穷凶极恶的刪庫代码。



这下原因是找到了，反引号应该改成**双引号**才对。

小林你真菜呀，那么简单的赋值命令都写错。

哈哈，确实菜，都说了嘛，当时快五一了，小林是飘着写这份代码的。

所以习惯性开启程序员内容的第一大武功：`ctrl+c` 和 `ctrl+v`。

把第一条赋值 `lic_path=`pwd`` 语句，复制粘贴了，然后只改了变量名，没注意反引号要修改成双引号，所以造成了删库的悲剧。

保留案发现场

既然发生了删库的事情，千万不要重启服务器，也不要关闭 ssh 连接的会话，而是要保留案发现场，接着查查还剩什么。

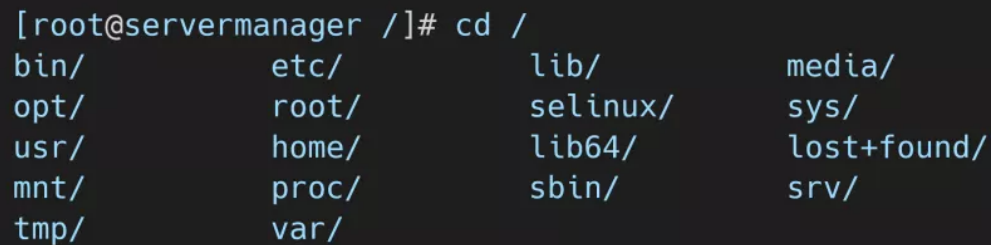
小林，这不是吹大炮嘛？`ls` 都没了，还怎么查？

还好这次是比较幸运，因为在执行脚本的时候，第一时间发现不对劲，**立马掐断**了还在运行的脚本，所以并非 Linux 所有文件都被删除了。

只要我掐的快，`rm -fr /*` 就干不死我。

虽然 `ls` 被删了，但所幸发现 `cd` 命令还能用。

只要 `cd` 用的好，它也能用出的 `ls` 效果。很简单，只需 `cd + Tab` 键就会自动出现指定目录下的所有文件。



```
[root@servermanager ~]# cd /  
bin/      etc/      lib/      media/  
opt/      root/     selinux/  sys/  
usr/      home/     lib64/    lost+found/  
mnt/      proc/     sbin/     srv/  
tmp/      var/
```

有了 `cd + Tab` 键，我们就可以查看每个目录下的文件，于是就可以一步一步来确认哪些系统文件被删了。

通过一番的确认和对比后，发现主要被删除的有四个目录分别是

- `/bin` 、 `/boot` 、 `/dev` 这三个目录整个都被删除了
- `/lib` 目录里的动态库部分被删除

来复习下上面这四个目录主要是存放了什么：

- `/bin` 存放常用系统命令，`ls`、`cp`、`rm`、`chmod` 等常用命令都在此目录；
- `/boot` 系统启动目录，保存与系统启动相关的文件，如内核文件和启动引导程序；

- `/dev` 设备文件保存位置；
- `/lib` 存放程序所需的动态库和静态库文件；

`/boot` 都被删除了，还好小林没有重启服务器，要是重启了服务器，就完犊子了，系统肯定起不来了。

`cd` 命令是在 `/sbin` 目录下，`/sbin` 还健全，所以 `cd` 是可以正常使用。

所幸重要的数据库信息和文件都还没删除，所以小林首要的目标是要恢复 `/bin`、`/boot`、`/dev`、`/lib` 这四个目录。

还原文件

由于 `/bin` 目录和 `/lib` 部分动态文件被删除，常用的传递文件的方式是无法使用的，如 `ftp`、`scp`、`mount` 等。

小林摸索了很久，竟然发现 `wget` 可以使用，`wget` 命令是在 `/usr/bin` 目录，所幸 `/usr/bin` 还健全。

于是，用了取巧的方法，先另一台正常的服务器，把 `/bin` 目录放到了 `Web` 服务器的 `Web` 目录，接着通过 `wget` 进行下载。

```
[root@servermanager ~]# wget http://10.200.1.49/source/bin/ls
--2020-04-29 09:18:11-- http://10.200.1.49/source/bin/ls
Connecting to 10.200.1.49:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 117048 (114K) [text/plain]
Saving to: "ls"

100%[=====>] 117,048 --.-K/s in 0.05s

2020-04-29 09:18:11 (2.21 MB/s) - "ls" saved [117048/117048]
```

有戏，看到了成功的曙光。

但是新的问题就来了，我下载过来的命令文件，是没有执行权限的。

```
[root@servermanager /]# /ls  
-bash: /ls: Permission denied
```

而 `chmod` 命令是在 `/bin` 目录的，它同样也被删除了，无法使用它来给予文件权限。

还在，在网上搜到了一个伟大命令 `perl`，可以通过它来给予文件权限：

```
perl -e "chmod 777, 'ls'"
```

真是个神奇的命令。

好了，这下赋值权限问题也解决了，成功在望了。

`wget` 是无法直接把 `/bin` 目录下载下来的，只能下载一个文件。

但是小林我不可能一个一个去下载来进行恢复，这得要何年何月才能完成。。。

小林就想到了一个方法：

- 先通过 `wget` 的方式下载 `tar` 命令，并通过 `perl` 给予 `tar` 命令权限
- 接着把另一台服务器把 `/bin` 目录打包成压缩文件，然后通过 `wget` 下载 `bin` 目录的压缩包文件
- 最后通过 `tar` 命令把 `bin` 压缩包解压出来

`/bin` 就这样恢复回来啦，剩余的其他目录 也是通过同样的操作恢复了回来。

小林的笑容渐渐恢复了回来，哈哈哈哈哈哈哈哈哈哈



遇到 `rm -fr /*` 删库事件发生，一定要沉住气，稳住心态

本次删库事件，之所以小林能幸运的恢复回来，有非常关键两点：

- 小林发现脚本执行不正常，果断立马的掐断它，没有造成重要的数据库信息被删除，如果掐断的时候再晚一点，可能就真没了。
- 小林发现常用命令无法使用的时候，没有重启服务器，不然服务器就起不来了，也没有关闭 `ssh` 会话，不然无法在重新连接 `ssh` 会话了，也就无法进行操作了。

如果以上两点都没做好，服务器恢复的难度就加大了很多，更严重的是五一节就没了的过了。



预防误执行 `rm -fr /*`

既然 `rm -fr /*` 是残忍的凶器，那么预防它是很有必要的，接下来跟大家讨论讨论预防它的几种方案。

方案一：rm -rf 删除目录时要判断目录

```
#!/bin/bash

work_path=`pwd`

#如果目录不为空，才执行删除操作
if [ ${work_path} != "" ];then
    rm -fr ${work_path}/*
fi
```

在执行删除目录操作前，先判断要删除的目录是否为空，不为空才执行删除操作。

方案二：Shell 脚本指定 `set -u`

执行脚本的时候，如果遇到不存在的变量，Bash 默认忽略它。

```
#!/bin/bash  
  
echo $a  
echo hello
```

上面代码中，`$a` 是一个不存在的变量，执行结果如下。

```
$ bash test.sh  
  
hello
```

可以发现，`echo $a` 输出了一个空行，`Bash` 忽略了不存在的 `$a`，然后继续执行 `echo hello`。

最好是遇到变量不存在，脚本应该报错，而不是一声不响地往下执行。

`set -u` 就用来改变这种行为，在脚本加上它，遇到不存在的变量就会报错，并停止执行。

```
#!/bin/bash  
set -u  
  
rm -fr $a/*  
echo hello
```

运行结果如下：

```
$ bash test.sh
test.sh: line 4: a: unbound variable
```

可以看到，因为 `a` 是未定义变量，脚本报错了，并且不再执行后面的语句。

方案三：safe-rm 替换 rm

`safe-rm` 是一个开源软件工具，这名字听起来就很安全嘛，所以它是用来替代不太安全的 `rm`。

它可以在 `/etc/safe-rm.conf` 中配置路径黑名单，定义哪些不能被 `safe-rm` 删除。

我们可以将 `safe-rm` 更名为 `rm`，假设定义了 `/etc/` 无能被删除，那么删除 `/etc` 时就会报错：

```
$ rm -rf /etc/
safe-rm: skipping /etc/
```

方案四：建立回收站机制

Windows 是有回收站的，即使误删了，也可以在回收站恢复。

所以，我们也可以在 Linux 实现回收站的机制。

实现思路：

删除文件时，它并不真正执行删除操作，而是将文件移动到一个特定目录，可以设置定时清楚回收站，或者在回收站里面的文件大小达到一定容量时（或者用时间做判断）执行删除操作以腾出空间。

可以写个 Shell 脚本替换 `rm` 命令，或者在需要删除文件的时候使用 `mv` 命令将文件移动到回收站。

① 创建回收站目录

```
mkdir /home/.trash
```

② 编写 `remove.sh` 脚本，内容如下


```
TRASH_DIR="/home/.trash"

for i in $*; do
    STAMP=`date +%s`
    fileName=`basename $i`

    #将对应文件 mv 至 .trash 目录
    mv $i ${TRASH_DIR}/${fileName}.${STAMP}
done
```

③ 修改 `~/.bashrc` , 用我们自建的 `remove.sh` 替代 `rm` 命令

```
alias rm="sh /home/remove.sh"
```

④ 设置 `crontab` , 定期清空垃圾箱, 如每天 0 点清空垃圾箱:

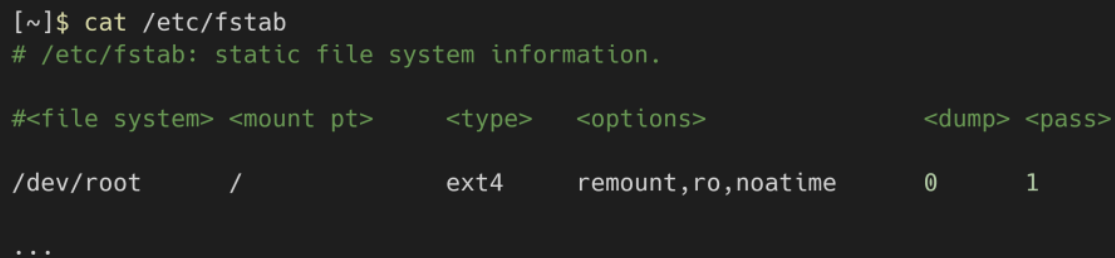
```
0 0 * * * rm -rf /home/.trash/*
```

⑤ 最后，执行以下命令，使之生效

```
source ~/.bashrc
```

方案五：根文件挂载成只读

在 `/etc/fstab` 文件，把 `/` 文件系统挂载成只读的方式。

A terminal window with a dark background and light green text. It shows the command 'cat /etc/fstab' and its output. The output is a table-like structure for the /etc/fstab file, showing the root filesystem mounted as ext4 with remount,ro,noatime options.

```
[~]$ cat /etc/fstab
# /etc/fstab: static file system information.

#<file system> <mount pt>      <type>      <options>          <dump> <pass>

/dev/root      /                ext4         remount,ro,noatime 0        1

...
```

其中 `remount,ro`，就表示只读的方式挂载。

只读的方式挂载后，进行删除操作是无法成功的：

```
[~]$ rm /bin -fr
rm: can't remove '/bin/fsync': Read-only file system
rm: can't remove '/bin/spawn-fcgi': Read-only file system
rm: can't remove '/bin/ntupdate': Read-only file system
rm: can't remove '/bin/monit': Read-only file system
rm: can't remove '/bin/ls': Read-only file system
rm: can't remove '/bin/df': Read-only file system
rm: can't remove '/bin/snmpgetnext': Read-only file system
rm: can't remove '/bin/stty': Read-only file system
rm: can't remove '/bin/dmesg': Read-only file system
rm: can't remove '/bin/ionice': Read-only file system
rm: can't remove '/bin/egrep': Read-only file system
rm: can't remove '/bin/cat': Read-only file system
rm: can't remove '/bin/chown': Read-only file system
rm: can't remove '/bin/autotest': Read-only file system
rm: can't remove '/bin/csfdebug': Read-only file system
rm: can't remove '/bin/netstat': Read-only file system
rm: can't remove '/bin/makemime': Read-only file system
rm: can't remove '/bin/ping': Read-only file system
rm: can't remove '/bin/rmdir': Read-only file system
rm: can't remove '/bin/fdflush': Read-only file system
rm: can't remove '/bin/mknod': Read-only file system
rm: can't remove '/bin/touch': Read-only file system
```

事后反思

涉及到 `rm -fr` 命令的代码，要留个心眼，要反复检查，要做好预防误执行 `rm -fr /*`，并在测试机验证完后，再拖到实体机上跑，千万不可大意。

就算的发生了 `rm -fr /*`，要第一时间停掉它，并且要做到三不要：

- 不要慌，不要心跳爆炸（稳住稳住）
- 不要隐瞒删库事件（不丢人）
- 不要重启服务器或断开 ssh 会话（保留现场）

只要立马掐断 `rm -fr /*`，它是干不死我们的。

利用当下环境剩有的命令，冷静分析，是有机会恢复的。

经过这个事情后，小林收获了一个 title：「[一个删过库没跑路的男人](#)」

酷吧！大家千万不要向我学习哦🐱

[阅读原文](#)

喜欢此内容的人还喜欢

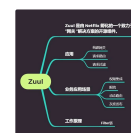
看书的一点小建议！

小林coding



19 張圖概覽Spring Cloud

JAVA高級架構



《HelloGitHub》第61 期

HelloGitHub

