

Linux磁盘设备文件(sda,sdb,sdc...)变化问题

在Linux下往往会碰到这样的问题，磁盘的设备文件，比如/dev/sda, sdb, sdc等等在某些情况下会混乱掉，比如sda变成了sdb或者sdc变成了sdb等等，这样无形会导致磁盘设备管理的混乱，最常见的比如Linux文件系统的启动问题。很多人在遇到这种问题的时候都去找磁盘、阵列厂家，怀疑是他们的的问题，其实这种底层的磁盘（单个磁盘或者RAID阵列）和Linux下磁盘设备文件的映射并不是磁盘、阵列厂家来决定的，而是Linux内核自身的原因。

目前Linux内核对于这种磁盘设备的映射基本上取决于三个顺序，一是磁盘驱动程序的加载；二是主机PCI插槽的监测；三是磁盘本身的监测，先来的当然是a，以此类推。所以，在出现热插拔了某些设备、重启等特殊情况下，实际磁盘在Linux下映射的设备文件可能由于这种“排队”的原因而发生改变，而这种底层“偷偷的”变化有时候会让管理员犯一些低级错误。

这是Linux Kernel的限制，所以目前还没办法来正面的克服应对，但有两个“迂回战术”的办法来减少可能出现的问题，一个是采用UUID设备唯一识别的方法，另一个是采用对设备卷做Label标识的办法。

一、UUID (globally unique identifier)，唯一的身份识别，是采用SCSI Inquiry命令的Page 83信息来映射磁盘设备的。例如我们可以在Linux下查询一些磁盘设备的UUID标识代码。

```
bash# ls -la /dev/disk/by-id
total 0
drwxr-xr-x 2 root root 280 Mar 11 12:29 .
drwxr-xr-x 5 root root 100 Mar 11 12:28 ..
lrwxrwxrwx 1 root root  9 Mar 11 12:29 edd-int13_dev80 -> ../../sda
lrwxrwxrwx 1 root root 10 Mar 11 12:29 edd-int13_dev80-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Mar 11 12:29 edd-int13_dev80-part3 -> ../../sda3
lrwxrwxrwx 1 root root 10 Mar 11 12:29 edd-int13_dev80-part4 -> ../../sda4
lrwxrwxrwx 1 root root 10 Mar 11 12:29 edd-int13_dev80-part5 -> ../../sda5
lrwxrwxrwx 1 root root 10 Mar 11 12:29 edd-int13_dev80-part6 -> ../../sda6
lrwxrwxrwx 1 root root  9 Mar 11 12:28 scsi-3600050e03d7c67007bf400009f890000 -
> ../../sda
lrwxrwxrwx 1 root root 10 Mar 11 12:28 scsi-3600050e03d7c67007bf400009f890000-p
art1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Mar 11 12:28 scsi-3600050e03d7c67007bf400009f890000-p
art3 -> ../../sda3
lrwxrwxrwx 1 root root 10 Mar 11 12:28 scsi-3600050e03d7c67007bf400009f890000-p
art4 -> ../../sda4
```

```
lrwxrwxrwx 1 root root 10 Mar 11 12:28 scsi-3600050e03d7c67007bf400009f890000-p
art5 -> ../../sda5
lrwxrwxrwx 1 root root 10 Mar 11 12:28 scsi-3600050e03d7c67007bf400009f890000-p
art6 -> ../../sda6
```

找到了磁盘设备唯一的UUID代码后，就可以加到/etc/grub.conf和/etc/fstab中，这样即使初始的系统盘sda变成了sdb，但Linux和文件系统的启动加载都是按照UUID来的，所以上层也不会受到影响，例如，

在 /etc/grub.conf系统启动入口中做以下更改：

```
kernel /boot/vmlinuz-2.6.27.7 root=/dev/disk/by-id/scsi-
3600050e03d7c67007bf400009f890000-part1
```

在 /etc/fstab文件系统启动入口中做以下更改：

```
/dev/disk/by-id/scsi-3600050e03d7c67007bf400009f890000-part1 /          ext3    1 1
/dev/disk/by-id/scsi-234892819987c8f828473829becf38289-part2 /home    ext3    1 1
```

二、第二种算是比较老式的解决方法，即对磁盘卷设置Label标签，同样的道理，系统启动的时候只看标签，不看底层的sda/sdb等设备号，所以也不会影响到系统、文件系统的启动。例如，

使用e2label命令对sda1和sdb1设置标签：

```
/sbin/e2label /dev/sda1 myroot
/sbin/e2label /dev/sdb1 myhome
```

之后在 /etc/grub.conf 系统启动入口中做以下更改：

```
kernel /boot/vmlinuz-2.6.29 ro root=LABEL=myroot
```

在 /etc/fstab文件系统启动入口中做以下更改：

```
LABEL=myroot      /          ext3 defaults 1 1  
LABEL=myhome /home ext3 defaults 1 1
```

当然，以上两种都是为了不影响系统和文件系统的启动采用的变通方法，在实际的系统管理时还是要密切注意底层设备的变化，否则如果出现了磁盘分区误删除的事情罪过可就大了。

分类: [linux](#) , [unix](#)

linux磁盘盘符漂移

1、Linux硬盘盘符分配原则

在Linux系统中，若存在多块硬盘，内核分配盘符的顺序是/dev/sda、/dev/sdb、/dev/sdc 在系统启动过程中，内核会按照扫描到硬盘的顺序分配盘符。

内核中分配盘符函数，见链接<http://ilinuxkernel.com/?p=794>

Linux内核通过IDR (integer ID) 整数ID管理机制来分配盘符，即找到一个空闲的整数。内核针对SCSI盘符，从0开始分配整数。整数0对应的盘符为/dev/sda，关系如下：

若index=0，则分配给此块SCSI硬盘的盘符为sda；

若index=1，则分配给此块SCSI硬盘的盘符为sdb；

... ..

若index=25，则分配给此块SCSI硬盘的盘符为sdz；

如有12硬盘的服务器中，每个物理槽位均插一块硬盘，且没有做RAID，硬盘物理槽位关系如下：

0	3	6	9
1	4	7	10
2	5	8	11

Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk

那么Linux内核启动后，IDR机制分配的盘符的整数为0、1、2、3、4、5、6、7、8、9、10、11，对应的硬盘盘符如下：

/dev/sd a	/dev/sd d	/dev/sd g	/dev/sd j
/dev/sd b	/dev/sd e	/dev/sd h	/dev/sd k
/dev/sd c	/dev/sd f	/dev/sd i	/dev/sd l

系统运行过程中，若拔掉第5块盘，如下图所示：

Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk		Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk

则此时系统中，/dev/sde盘符消失，内核针对SCSI盘符的IDR整数4就空闲出来。当再次将该硬盘插入时，得到的盘符仍然是/dev/sde。

2、非热插拔硬盘盘符分配示例

如下图，第5个硬盘物理槽位没有硬盘。系统重启后，盘符对应关系如下。系统中没有盘符/dev/sdl，缺少不是最后一块硬盘，而是第5块硬盘。

Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk		Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk

/dev/sd a	/dev/sd d	/dev/sd f	/dev/sd i
/dev/sd b		/dev/sd g	/dev/sd j
/dev/sd c	/dev/sd e	/dev/sd h	/dev/sd k

下图是第5块和第9块物理槽位硬盘不在位时，OS盘符对应关系。

Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk		Hard Disk	Hard Disk
Hard Disk	Hard Disk		Hard Disk

/dev/sd a	/dev/sd d	/dev/sd f	/dev/sd h
/dev/sd b		/dev/sd g	/dev/sd i
/dev/sd c	/dev/sd e		/dev/sd j

从这我们可以看出：

- 1) 非热插拔的磁盘它盘符是按照整数ID来分配
- 2) 由于是非热插拔磁盘不能在通电的时候插入磁盘

3、热插拔硬盘盘符分配示例

下图是第1、5、9块硬盘不在位时，OS盘符对应关系，此时没有热插拔硬盘。

	Hard Disk	Hard Disk	Hard Disk
Hard Disk		Hard Disk	Hard Disk
Hard Disk	Hard Disk		Hard Disk

	/dev/sd c	/dev/sd e	/dev/sd g
/dev/sd a		/dev/sd f	/dev/sd h
/dev/sd b	/dev/sd d		/dev/sd i

系统运行过程中，先热插拔一块到第1个物理槽位，此时得到的盘符为/dev/sdj，然后插入一块硬盘到第9个物理槽位，此时分配的盘符为/dev/sdk。

Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk		Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk

/dev/sd j	/dev/sd c	/dev/sd e	/dev/sd g
/dev/sd a		/dev/sd f	/dev/sd h
/dev/sd b	/dev/sd d	/dev/sd k	/dev/sd i

若插入2块硬盘后，机器重启，则盘符对应关系可能如下所示。从这里我们可以看出，硬盘盘符和物理槽位没有必然直接对应关系。

Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk		Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk

/dev/sd a	/dev/sd d	/dev/sd f	/dev/sd i
/dev/sd b		/dev/sd g	/dev/sd j
/dev/sd c	/dev/sd e	/dev/sd h	/dev/sd k

从这我们可以看出

- 1) 由于磁盘是热插拔的可以在通电的时候插入磁盘
- 2) 新插入的磁盘盘符是按照整数ID顺序来进行分配的
- 3) 建议平时我们不要随意挪动硬盘，避免系统盘和数据盘调混了

4、硬盘盘符为什么会漂移

服务器只有12块物理硬盘，但在系统运行过程中或更换硬盘时，会出现/dev/sdm、/dev/sdn等类似盘符。

下面示例，系统运行过程中，我们把第5块硬盘拔出再插入，此时得到的硬盘盘符可能为/dev/sdm。

Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk
Hard Disk	Hard Disk	Hard Disk	Hard Disk

/dev/sd a	/dev/sd d	/dev/sd g	/dev/sd j
/dev/sd b	/dev/sd m	/dev/sd h	/dev/sd k
/dev/sd c	/dev/sd f	/dev/sd i	/dev/sd l

当我们拔出硬盘后，内核会调用sd_remove() 函数卸载硬盘，正常情况下会清除该硬盘盘符占用的所有资源，包括SCSI盘符对应的IDR整数。但我们拔出硬盘时，若系统有进程正在访问该硬盘，则内核不会删除对应IDR的整数，该整数就会被占用，再次插入硬盘时，就分配新的IDR整数给盘符，导致盘符漂移。

如上面第5块硬盘，内核分配的IDR值为4，但硬盘拔出再插入后，应为IDR值为4没被释放，内核找到空闲的IDR就会12，此时盘符就变为/dev/sdm。盘符变为/dev/sdn等时，以此类推

参考链接：<https://ilinuxkernel.com/?p=958>

posted on 2021-03-02 20:46 gentleman_hai 阅读(4202) 评论(0) 编辑 收藏 举报

Linux内核SCSI硬盘盘符分配

Jul30

2011 (<http://ilinuxkernel.com/?p=794>)_Written by [chen](http://ilinuxkernel.com/?author=1) (<http://ilinuxkernel.com/?author=1>)

我们以Redhat Enterprise Linux 6内核源码2.6.32-71.el6版本为例，分析Linux内核SCSI层是如何给硬盘分配盘符的，即/dev/sda、/dev/sdb ... /dev/sdm等盘符的由来。

sd_probe () 函数

系统中有新的SCSI磁盘（包括USB硬盘）插入，就会调用sd_probe () 函数。

哪里决定盘符？

index的值决定了盘符（sd_probe () 函数中第2277行）。

若index=0，则分配给此块SCSI硬盘的盘符为sda；

若index=1，则分配给此块SCSI硬盘的盘符为sdb；

... ..

若index=25，则分配给此块SCSI硬盘的盘符为sdz；

函数在文件drivers/scsi/sd.c中。

```
02248: static int sd_probe(struct device *dev)
02249: {
02250:     struct scsi_device *sdp = to_scsi_device(dev);
02251:     struct scsi_disk *sdkp;
02252:     struct gendisk *gd;
02253:     u32 index;
02254:     int error;
02255:
02256:     error = -ENODEV;
02257:     if (sdp->type != TYPE_DISK && sdp->type != TYPE_MOD && sdp->type !=
02257: TYPE_RBC)
02258:         goto out;
02259:
02260:     SCSI_LOG_HLQUEUE(3, sdev_printk(KERN_INFO, sdp,
02261:         "sd_attach\n"));
```

```

02262:
02263: error=-ENOMEM;
02264: sdkp=kzalloc(sizeof(*sdkp),GFP_KERNEL);
02265: if(!sdkp)
02266:     goto↓out;
02267:
02268: gd=alloc_disk(SD_MINORS);
02269: if(!gd)
02270:     goto↓out_free;
02271:
02272: do{
02273:     if(!ida_pre_get(&sd_index_ida,GFP_KERNEL))
02274:         goto↓out_put;
02275:
02276:     spin_lock(&sd_index_lock);
02277:     error=ida_get_new(&sd_index_ida,&index);
02278:     spin_unlock(&sd_index_lock);
02279: } while(error==EAGAIN);
02280:
02281: if(error)
02282:     goto↓out_put;
02283:
02284: error=sd_format_disk_name("sd",index,gd->disk_name,DISK_NAME_LEN);
02285: if(error)
02286:     goto↓out_free_index;
02287:
02288: sdkp->device= sdp;
02289: sdkp->driver= &sd_template;
02290: sdkp->disk= gd;
02291: sdkp->index=index;
02292: sdkp->openers= 0;
02293: sdkp->previous_state= 1;
02294:
02295: if(!sdp->request_queue->rq_timeout){
02296:     if(sdp->type !=TYPE_MOD)
02297:         blk_queue_rq_timeout(sdp->request_queue,SD_TIMEOUT);
02298:     else
02299:         blk_queue_rq_timeout(sdp->request_queue,

```

```

02300:                SD_MOD_TIMEOUT);
02301:    }
02302:
02303:    device_initialize(&sdkp->dev);
02304:    sdkp->dev.parent= &sdp->sdev_gendev;
02305:    sdkp->dev.class=&sd_disk_class;
02306:    dev_set_name(&sdkp->dev,dev_name(&sdp->sdev_gendev));
02307:
02308:    if(device_add(&sdkp->dev))
02309:        goto↓out_free_index;
02310:
02311:    get_device(&sdp->sdev_gendev);
02312:
02313:    get_device(&sdkp->dev); /*preventreleasebeforeasync_schedule*/
02314:    async_schedule(sd_probe_async,sdkp);
02315:
02316:    return 0;

```

Posted in I/O系统 (<http://ilinuxkernel.com/?cat=7>), 内核基础 (<http://ilinuxkernel.com/?cat=3>)