Linux盘符绑定实现原理

http://www.ilinuxkernel.com

正一

2016.7.25

目 录

Linux盘符的分配

❑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？

```
01438:      spin_lock(&sd_index_lock );
01439:      error = idr_get_new(&sd_index_idr, NULL, &index);
01440:      spin_unlock(&sd_index_lock );
01441:
```

index的值决定了盘符。

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；

Linux盘符的分配

```
01408: static int sd_probe(struct device *dev)
01409: {
  ... ...
01435:         if (! idr_pre_get(&sd_index_idr, GFP_KERNEL))
01436:             goto ↓out_put;
01437:
01438:         spin_lock(&sd_index_lock );
01439:         error = idr_get_new(&sd_index_idr, NULL, &index);
01440:         spin_unlock(&sd_index_lock );
01441:
01442:         if (index >= SD_MAX_DISKS)
01443:             error = - EBUSY;
01444:         if (error)
01445:             goto ↓out_put;
    ... ...
01465:         if (index < 26) {
01466:             sprintf(gd- >disk_name, "sd%c", 'a' + index % 26);
01467:         } else if (index < (26 + 1) * 26) {
01468:             sprintf(gd- >disk_name, "sd%c%c",
01469:                 'a' + index / 26 - 1,'a' + index % 26);
01470:         } else {
01471:             const unsigned int m1 = (index / 26 - 1) / 26 - 1;
01472:             const unsigned int m2 = (index / 26 - 1) % 26;
01473:             const unsigned int m3 =  index % 26;
01474:             sprintf(gd- >disk_name, "sd%c%c%c",
01475:                     'a' + m1, 'a' + m2, 'a' + m3);
01476:         }
        ... ...
01506: } ? end sd_probe ?
```

Linux盘符的分配

❑index值的由来

```
01435:      if (! idr_pre_get(&sd_index_idr, GFP_KERNEL))
01436:      goto ↓out_put;
01437:
01438:      spin_lock(&sd_index_lock );
01439:      error = idr_get_new(&sd_index_idr, NULL, &index);
01440:      spin_unlock(&sd_index_lock );
```

index的值调用idr_get_new（）函数获取。

Linux内核IDR机制

❑idr的由来

如何通过设备的ID，来找到快速设备相应的数据结构?

❑内核idr机制

- idr在linux内核中指的就是整数ID管理机制，从本质上来说，这就是一种将整数ID号和特定指针关联在一起的机制。
- idr机制适用在那些需要把某个整数和特定指针关联在一起的地方。
- 该机制内部采用radix树实现，可以很方便地将整数和指针关联起来，并且具有很高的搜索效率。

Linux内核IDR机制

❑通过idr机制获取ID过程

（1）为idr分配内存

int ***idr_pre_get****(struct idr *idp, unsigned int gfp_mask);*

每次通过idr获得ID号之前，需要先分配内存。

返回0表示错误，非零值代表正常。

（2）分配ID号并将ID号和指针关联

int ***idr_get_new****(struct idr *idp, void *ptr, int *id);*

int ***idr_get_new_above****(struct idr *idp, void *ptr, int start_id, int *id);*

idp: 之前通过idr_init初始化的idr指针

id: 由内核自动分配的ID号

ptr: 和ID号相关联的指针

start_id: 起始ID号。内核在分配ID号时，会从start_id开始。

Linux内核IDR机制

```
00261: int  idr_get_new(struct idr *idp, void *ptr, int *id)
00262: {
00263:         int rv;
00264:         rv = idr_get_new_above_int(idp, ptr, 0);
00265:         / *
00266:         * This is a cheap hack until the IDR code can be fixed to
00267:         * return proper error values.
00268:         */
00269:         if (rv < 0) {
00270:         if (rv == - 1)
00271:                 return - EAGAIN;
00272:         else / * Will be - 3 */
00273:                 return - ENOSPC;
00274:         }
00275:         *id = rv;
00276:             return 0;
00277: }
00278:  EXPORT_SYMBOL(idr_get_new);
00279:
```

Linux盘符绑定

❑如何控制盘符的名称？

控制index的值即可。

❑如何做到盘符绑定？

做到硬盘所在的槽位ID与index值绑定即可。

即若

硬盘槽位ID为1，则分配的index值为0，得到的盘符为sda，

硬盘槽位ID为2，则分配的index值为1，得到的盘符为sdb，

……

硬盘槽位ID为12，则分配的index值为11，得到的盘符为sdl，

……

Linux盘符绑定

❑如何获取硬盘槽位ID？

只有磁盘控制器才知道某块硬盘所在的物理槽位号，所以借助磁盘控制器驱动，就可获取硬盘所在的槽位ID。

本材料以LSI控制器及对应的mpt驱动为例，介绍获取硬盘所在槽位号的过程。

❑热插拔硬盘时所在槽位ID获取

热插拔硬盘时，会调用mptsas_hotplug_work(void *arg)函数。

Linux盘符绑定

❑热插拔时从哪里知道硬盘槽位ID？

热插拔硬盘时，message frame消息中记录了发生热插拔事件所在的物理槽位ID。而驱动将这个物理槽位ID存放在mptsas_hotplug_event数据结构中的phy_id成员变量中。

```
00072: struct  mptsas_hotplug_event {
00073:          struct work_struct hotplug_work;
00074:          MPT_ADAPTER          *ioc;
00075:          enum mptsas_hotplug_action event_type;
00076:          u64          sas_address;
00077:          u8           channel;
00078:          u8           id;
00079:          u32          device_info;
00080:          u16          handle;
00081:          u8           phy_id;
00082:          u8           phys_disk_num;      /* hrc - unique index*/
00083:          u8           retries;
00084:          u8           refresh_raid_config_pages;
00085:          struct scsi_device *sdev;
00086:          struct list_head          list;
00087: };
```

Linux盘符绑定

❑热插拔硬盘盘符绑定

知道了发生热插拔事件所在的物理槽位ID，就可以控制sd_probe（）函数中的index值，进而控制分配给该块硬盘的盘符。

❑系统启动过程中，无任何热插拔，如何做到盘符绑定？

RHEL4/SLES9和RHEL5/SLES10所使用的mpt驱动版本有所不同（详情请参考mptsas_probe函数）。

下面我们分别讨论RHEL4/SLES9和RHEL5/SLES10系列代码中系统启动过程中如何获取物理槽位ID。

Linux盘符绑定

❑RHEL5/SLES10系统启动过程中，硬盘物理槽位ID的获取

mptsas_probe（）调用的mptsas_scan_sas_topology（），进而调用mptsas_probe_expander_phys（）函数，进而调用mptsas_probe_one_phy（）函数。

```
00143: struct mptsas_devinfo {
00144:         u16 handle;          /* unique id to address this device */
00145:         u16 handle_parent;          /* unique id to address parent device */
00146:         u16 handle_enclosure; /* enclosure identifier of the enclosure */
00147:         u16 slot; /* physical slot in enclosure */
00148:         u8          phy_id;  /* phy number of parent device */
00149:         u8          port_id; /* sas physical port this device
00150:         is assoc'd with */
00151:         u8          id;          /* logical target id of this device */
00152:         u32 phys_disk_num; /* phys disk id, for csmi- ioctls */
00153:         u8          channel;/* logical bus number of this device */
00154:         u64 sas_address; /* WWN of this device,
00155:         SATA is assigned by HBA,expander */
00156:         u32 device_info; /* bitfield detailed info about this device */
00157: };
```

❑sd_probe()函数调用栈

Linux盘符绑定

[<f883b327>] sd_probe+0x33a/0x35b [sd_mod]
[<c055255e>] __device_attach+0x0/0x5
[<c055250e>] driver_probe_device+0x42/0x92
[<c0551e68>] bus_for_each_drv+0x37/0x5e
[<c05525aa>] device_attach+0x47/0x58
[<c055255e>] __device_attach+0x0/0x5
[<c0551bca>] bus_attach_device+0x13/0x26
[<c05510f2>] device_add+0x204/0x2de
[<f8862a9b>] scsi_sysfs_add_sdev+0x2a/0x1d2 [scsi_mod]
[<f8816548>] mptscsih_slave_configure+0xb2/0x13d [mptscsih]
[<f88614c4>] scsi_probe_and_add_lun+0x7d7/0x8c1 [scsi_mod]
[<f8861a9f>] __scsi_scan_target+0xb1/0x58c [scsi_mod]
[<c04a7ed1>] sysfs_new_dirent+0x53/0x5d
[<c04a7f25>] sysfs_make_dirent+0x10/0x6c
[<c04a773d>] sysfs_add_file+0x51/0x6a
[<c04a774c>] sysfs_add_file+0x60/0x6a
[<c04f72e8>] acpi_pci_find_root_bridge+0x1e/0x45
[<f88621c7>] scsi_scan_target+0x5f/0x73 [scsi_mod]
[<f884174f>] sas_rphy_add+0xd8/0xe3 [scsi_transport_sas]
[<f88acfc8>] **mptsas_probe_one_phy**+0x422/0x4de [mptsas]
[<f88ac9a7>] mptsas_setup_wide_ports+0x115/0x314 [mptsas]
[<f88add1e>] mptsas_probe_hba_phys+0x813/0x85e [mptsas]
[<f88be27d>] mpt_timer_expired+0x0/0x4e [mptbase]
[<f88be27d>] mpt_timer_expired+0x0/0x4e [mptbase]
[<f88be27d>] mpt_timer_expired+0x0/0x4e [mptbase]
[<f88aed82>] **mptsas_probe**+0x373/0x3fb [mptsas]

Linux盘符绑定

❑RHEL4/SLES9系统启动过程中，硬盘物理槽位ID的获取

mptsas_probe（）函数是向各个SCSI通道发送SCSI命令，若有响应则说明该槽位存在硬盘，最后调用sd_probe（）函数来分配盘符，产生新的磁盘设备。而在此过程中，无任何机会获取硬盘所在的物理槽位ID；包括mpt驱动，也没给出新发现的硬盘所在的物理槽位ID。

在mptsas_probe（）函数执行scsi_scan_host(sh)之前，我们可以借助mpt_sas_get_info（）函数来获取各个槽位的磁盘在位情况。这样我们就可以知道当前添加的硬盘所在物理槽位ID了。

Linux盘符绑定

❑盘符绑定源码何时会执行？

- **系统启动时，**加载mptsas驱动过程中，会扫描机器上所有的硬盘，对找到的每块硬盘，调用sd_probe（）函数，分配盘符，系统就可以使用。

- **当空闲的硬盘槽位有硬盘插入时，**也会调用sd_probe（）函数，分配盘符，将新硬盘添加到系统中。

- **硬盘读写过程中，不会执行任何相关盘符绑定源码。**内核SCSI模块和mpt驱动中的硬盘读写函数，未作任何修改。

❑服务器厂商认证源码何时会执行？

- 系统启动时只执行一次，后续不再执行。

- 硬盘读写过程中，不会执行服务器厂商验证代码。

Linux盘符绑定

❑盘符绑定是否会影响硬盘读写性能？

- 对硬盘读写性能，无任何影响!
- 盘符绑定源码只会与机器上的硬盘变化有关，与硬盘数据读写无关。
- 内核SCSI模块和mpt驱动中的硬盘读写函数，未作任何修改。
- 实现盘符绑定后的驱动和正常的驱动，**硬盘读写流程无任何差异，无任何额外执行代码。**

谢谢！