

落寞微蓝

做为一个普通人，只需做好三件事：看清这世界、修炼好自己、影响身边人。

目录视图 摘要视图 RSS 订阅

个人资料



luomoweilan

访问：148488次
积分：2196
等级：BLOG 5
排名：第13218名

原创：69篇
转载：45篇
译文：0篇
评论：5条

文章分类

- linux使用---系统管理 (7)
- linux使用---文件系统 (6)
- linux使用---网络 (0)
- linux编程---vim (6)
- linux编程---git (4)
- linux内核 (13)
- tmp (3)
- 其他 (35)

推荐文章

- * 程序员10月书讯, 评论得书
- * Android中Xposed框架篇---修改系统位置信息实现自身隐藏功能
- * Chromium插件(Plugin)模块(Module)加载过程分析
- * Android TV开发总结--构建一个TV app的直播节目实例
- * 架构设计: 系统存储--MySQL简单主从方案及暴露的问题

文章存档

- 2016年01月 (1)
- 2015年11月 (1)
- 2014年11月 (3)
- 2014年10月 (6)
- 2014年09月 (4)
- 2014年04月 (5)
- 2014年03月 (5)
- 2014年02月 (4)
- 2014年01月 (5)

移动信息安全的漏洞和逆向原理 程序员11月书讯, 评论得书啦 Get IT技能知识库, 50个领域一键直达

linux文件系统初始化过程(2)---挂载rootfs文件系统

标签: linux VFS rootfs initrd cpio

2014-01-05 23:02 9025人阅读 评论(1) 收藏 举报

分类:

linux内核 (12)

版权声明: 本文为博主原创文章, 未经博主允许不得转载。

一、目的

本文主要讲述linux3.10文件系统初始化过程的第一阶段: 挂载rootfs文件系统。
rootfs是基于内存的文件系统, 所有操作都在内存中完成; 也没有实际的存储设备, 所以不需要设备驱动程序的参与。基于以上原因, linux在启动阶段使用rootfs文件系统, 当磁盘驱动程序和磁盘文件系统成功加载后, linux系统会将系统根目录从rootfs切换到磁盘文件系统。

二、主要函数调用过程

图1描述了挂载rootfs的函数调用关系(图中红色部分), 便于后面的分析。
从图中发现, 在挂载rootfs前会先挂载sysfs, 这样做的原因是确保sysfs能够完整的记录下设备驱动模型。
sysfs_init()完成注册和挂载sysfs文件系统的功能; init_rootfs()负责注册rootfs, init_mount_tree()负责挂载rootfs, 并将init_task的命名空间与之联系起来。

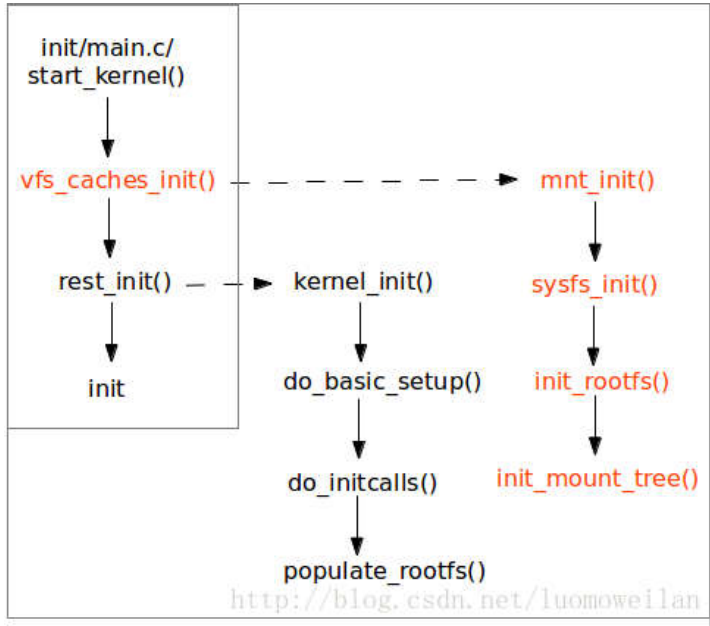


图1

三、linux文件系统初始化

vfs_cache_init()首先建立并初始化目录hash表dentry_hashtable和索引节点hash表inode_hashtable; 然后设置内核可以打开的最大文件数; 最后调用mnt_init()完成sysfs和

2013年01月 (2)
 2012年12月 (8)
 2012年02月 (1)
 2011年12月 (3)
 2011年07月 (2)
 2011年06月 (5)
 2011年03月 (7)
 2011年01月 (1)
 2010年12月 (3)
 2010年06月 (4)
 2010年04月 (1)
 2010年03月 (23)
 2009年09月 (1)
 2008年11月 (5)
 2008年10月 (14)

阅读排行

linux文件系统初始化过程 (9013)
 Vim的使用(三) (6512)
 Linux邮件列表的订阅与退订 (5484)
 Ping--127.0.0.1 (4842)
 linux进程管理(1)---进程控制 (4493)
 linux系统管理(4)---dpkg (3222)
 linux文件系统初始化过程 (2941)
 Linux启动代码header.S (2903)
 linux系统管理(3)---runlevel (2836)
 linux文件系统命令(5)---find (2805)

最新评论

linux进程管理(1)---进程描述符
 garyzhang2681: 总之, POSIX的进程ID就是tgid成员;POSIX的线程ID就是pid成员。这句话错了吧? 应该是...

linux系统管理(1)---man
 luomoweilan: @qq_25942591: 文中提到的pager特指man pager中的pager。

linux系统管理(1)---man
 qq_25942591: 楼主你好, 我觉得你对pager的解释有错误, 我输入"man man"之后, 里头有一段话。-P pa...

linux文件系统初始化过程(2)---挂载
 cherish_hao: 想请教一些作者, 上述的图是用什么软件画的? 感觉很清晰明了。

linux文件系统初始化过程(1)---概述
 hushup: 写得真好

linux系统管理(3)---runlevel
 mxgsgtc: mark! 解决了我的一個疑虑, 非常感謝!

rootfs文件系统的注册和挂载。

linux使用哈希表存储目录和索引节点, 以提高目录和索引节点的查找效率; dentry_hashtable是目录哈希表, inode_hashtable是索引节点哈希表。

四、挂载sysfs文件系统

sysfs用来记录和展示linux驱动模型, sysfs先于rootfs挂载是为全面展示linux驱动模型做好准备。

mnt_init()调用sysfs_init()注册并挂载sysfs文件系统, 然后调用kobject_create_and_add()创建“fs”目录。

```
2735 err = sysfs_init();
2736 if (err)
2737     printk(KERN_WARNING "%s: sysfs_init error: %d\n",
2738         __func__, err);
2739 fs_kobj = kobject_create_and_add("fs", NULL);
2740 if (!fs_kobj)
2741     printk(KERN_WARNING "%s: kobj create error\n", __func__);
```

下面详细介绍sysfs文件系统的挂载过程:

1、sysfs_init()调用register_filesystem()注册文件系统类型sysfs_fs_type, 并加入到全局单链表file_systems中。sysfs_fs_type定义如下, .mount成员函数负责超级块、根目录和索引节点的创建和初始化工作。

```
173     err = register_filesystem(&sysfs_fs_type);
174     if (!err) {
175         sysfs_mnt = kern_mount(&sysfs_fs_type);
176         if (IS_ERR(sysfs_mnt)) {
177             printk(KERN_ERR "sysfs: could not mount!\n");
178             err = PTR_ERR(sysfs_mnt);
179             sysfs_mnt = NULL;
180             unregister_filesystem(&sysfs_fs_type);
181             goto out_err;
182         }
183     }

152 static struct file_system_type sysfs_fs_type = {
153     .name        = "sysfs",
154     .mount       = sysfs_mount,
155     .kill_sb     = sysfs_kill_sb,
156     .fs_flags    = FS_USERNS_MOUNT,
157 };
```

2、sysfs_init()->kern_mount()->vfs_kern_mount()创建并初始化struct mount挂载点, 并使用全局变量sysfs_mnt保存该挂载点的挂载项(mnt成员)。

```
783     mnt = alloc_vfs_mnt(name);
784     if (!mnt)
785         return ERR_PTR(-ENOMEM);
```

3、kern_mount()调用sysfs_fs_type的.mount成员sysfs_mount()创建并初始化超级块、根目录'/'、根目录的索引节点等数据结构; 并且把超级块添加到全局单链表super_blocks中, 把索引节点添加到hash表inode_hashtable和超级块的inode链表中。

目前, 我们可以得出一个重要结论: kern_mount()主要完成挂载点、超级块、根目录和索引节点的创建和初始化操作, 可以看成是一个原子操作, 这个函数以后会频繁使用。

```
790     root = mount_fs(type, flags, name, data);
1091 struct dentry *
1092 mount_fs(struct file_system_type *type, int flags, const char *name, void *data)
1093 {
```

```

1094     struct dentry *root;
        ...
1108
1109     root = type->mount(type, flags, name, data);

107 static struct dentry *sysfs_mount(struct file_system_type *fs_type,
108     int flags, const char *dev_name, void *data)
109 {
        ...
112     struct super_block *sb;
        ...
125     sb = sget(fs_type, sysfs_test_super, sysfs_set_super, flags, info);
        ...
130     if (!sb->s_root) {
131         error = sysfs_fill_super(sb, data, flags & MS_SILENT ? 1 : 0);

```

4、vfs_kern_mount()初始化挂载点的根目录和超级块。

```

796 mnt->mnt.mnt_root = root;
797 mnt->mnt.mnt_sb = root->d_sb;
798 mnt->mnt_mountpoint = mnt->mnt.mnt_root;
799 mnt->mnt_parent = mnt;

```

5、mnt_init()调用kobject_create_and_add()创建“fs”目录。

通过以上步骤, sysfs文件系统在VFS中的视图如图2所示:挂载点指向超级块和根目录;超级块处在super_blocks单链表中, 并且链接起所有属于该文件系统的索引节点;根目录“/”和目录“fs”指向各自的索引节点;为了提高查找效率, 索引节点保存在hash表中。

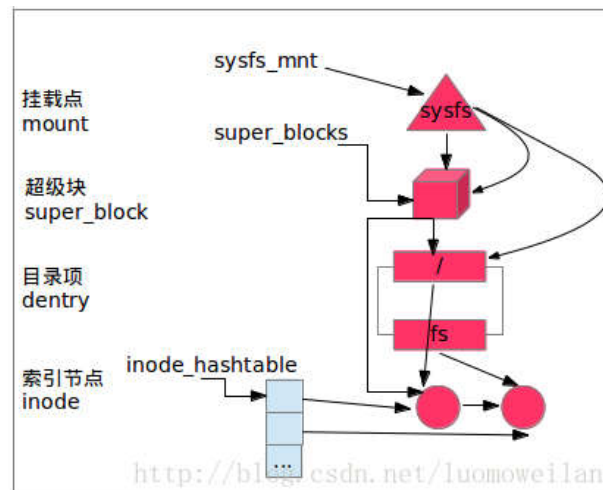


图2

五、挂载rootfs文件系统

mnt_init()调用init_rootfs()注册rootfs, 然后调用init_mount_tree()挂载rootfs。

下面详细介绍rootfs文件系统的挂载过程:

1、mnt_init()调用init_rootfs()注册文件系统类型rootfs_fs_type, 并加入到全局单链表file_systems中。

rootfs_fs_type定义如下, mount成员函数负责超级块、根目录和索引节点的建立和初始化工作。

```

265 static struct file_system_type rootfs_fs_type = {
266     .name      = "rootfs",
267     .mount     = rootfs_mount,
268     .kill_sb   = kill_litter_super,
269 };

```

2、init_mount_tree()调用vfs_kern_mount()挂载rootfs文件系统, 详细的挂载过程与sysfs文件系统类似, 不再赘述。

3、init_mount_tree()调用create_mnt_ns()创建命名空间，并设置该命名空间的挂载点为rootfs的挂载点，同时将rootfs的挂载点链接到该命名空间的双向链表中。

```

2459 static struct mnt_namespace *create_mnt_ns(struct vfsmount *m)
2460 {
2461     struct mnt_namespace *new_ns = alloc_mnt_ns(&init_user_ns);
2462     if (!IS_ERR(new_ns)) {
2463         struct mount *mnt = real_mount(m);
2464         mnt->mnt_ns = new_ns;
2465         new_ns->root = mnt;
2466         list_add(&mnt->mnt_list, &new_ns->list);
2467     }

```

4、init_mount_tree()设置init_task的命名空间，同时调用set_fs_pwd()和set_fs_root()设置init_task任务的当前目录和根目录为rootfs的根目录'/'。

```

2696     ns = create_mnt_ns(mnt);
2697     if (IS_ERR(ns))
2698         panic("Can't allocate initial namespace");
2699
2700     init_task.nsproxy->mnt_ns = ns;
2701     get_mnt_ns(ns);
2702
2703     root.mnt = mnt;
2704     root.dentry = mnt->mnt_root;
2705
2706     set_fs_pwd(current->fs, &root);
2707     set_fs_root(current->fs, &root);

```

通过以上分析，我们发现sysfs和rootfs的区别在于：虽然系统同时挂载了sysfs和rootfs文件系统，但是只有rootfs处于init_task进程的命名空间内，也就是说系统当前实际使用的是rootfs文件系统。

此时，sysfs和rootfs在VFS中的视图如图3所示：为了突出主要关系，省略了挂载点指向超级块和根目录。

从图中看出，rootfs处于进程的命名空间中，并且进程的fs_struct数据结构的root和pwd都指向了rootfs的根目录'/'，所以用户实际使用的是rootfs文件系统。另外，rootfs为VFS提供了'/'根目录，所以文件操作和文件系统的挂载操作都可以在VFS上进行了。

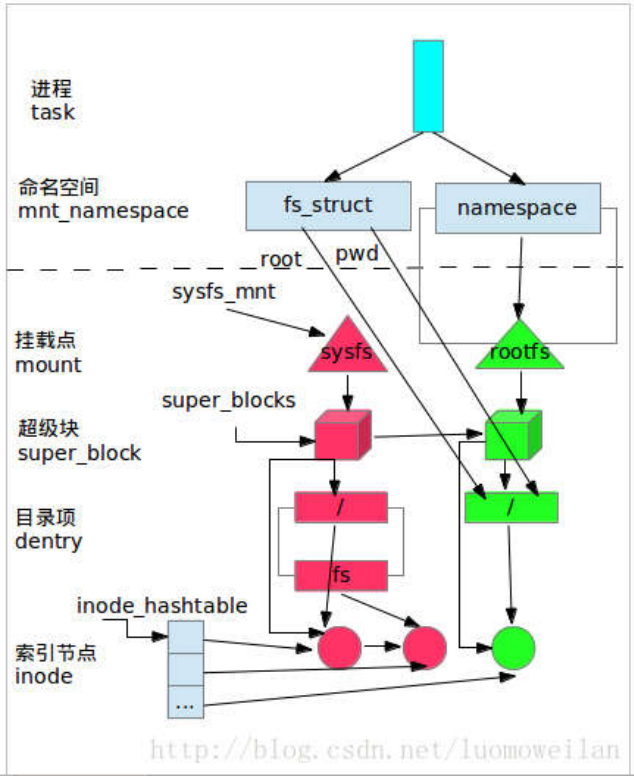


图3

六、总结

linux文件系统在初始化时, 同时挂载了sysfs和rootfs文件系统, 但是只有rootfs处于进程的命名空间中, 且进程的root目录和pwd目录都指向rootfs的根目录。至此, linux的VFS已经准备好了根目录(rootfs的根目录'/'), 此时用户可以使用系统调用对VFS树进行扩展。

版权声明:

原创作品, 如非商业性转载, 请注明出处;如商业性转载出版, 请与作者联系。

顶 2
踩 0

上一篇 linux文件系统初始化过程(1)---概述

下一篇 linux系统调用

猜你在找

查看评论

* 以上用户言论只代表其个人观点, 不代表CSDN网站的观点或立场