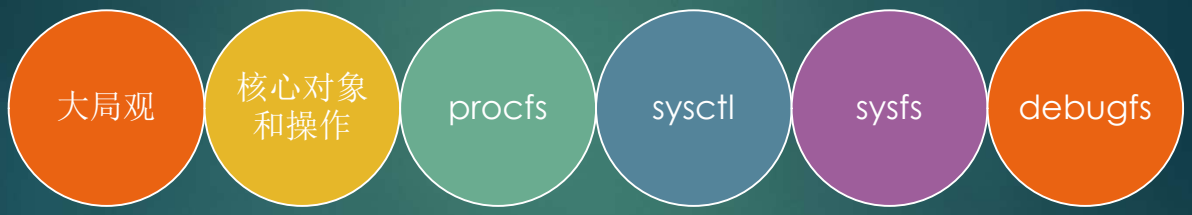
LINUX系统高级调试和优化

-- 虚拟文件系统(VFS)

张银奎

2016/9/11

1



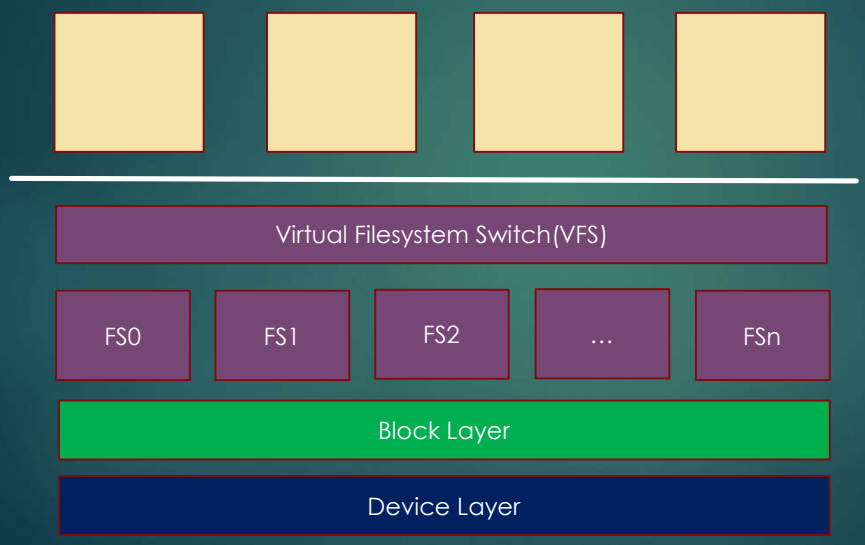
2

The Virtual File System (also known as the Virtual Filesystem Switch) is the software layer in the kernel that provides the filesystem interface to userspace programs. It also provides an abstraction within the kernel which allows different filesystem implementations to coexist.

/documentation/filesystems/vfs.txt

3

Layered Architecture

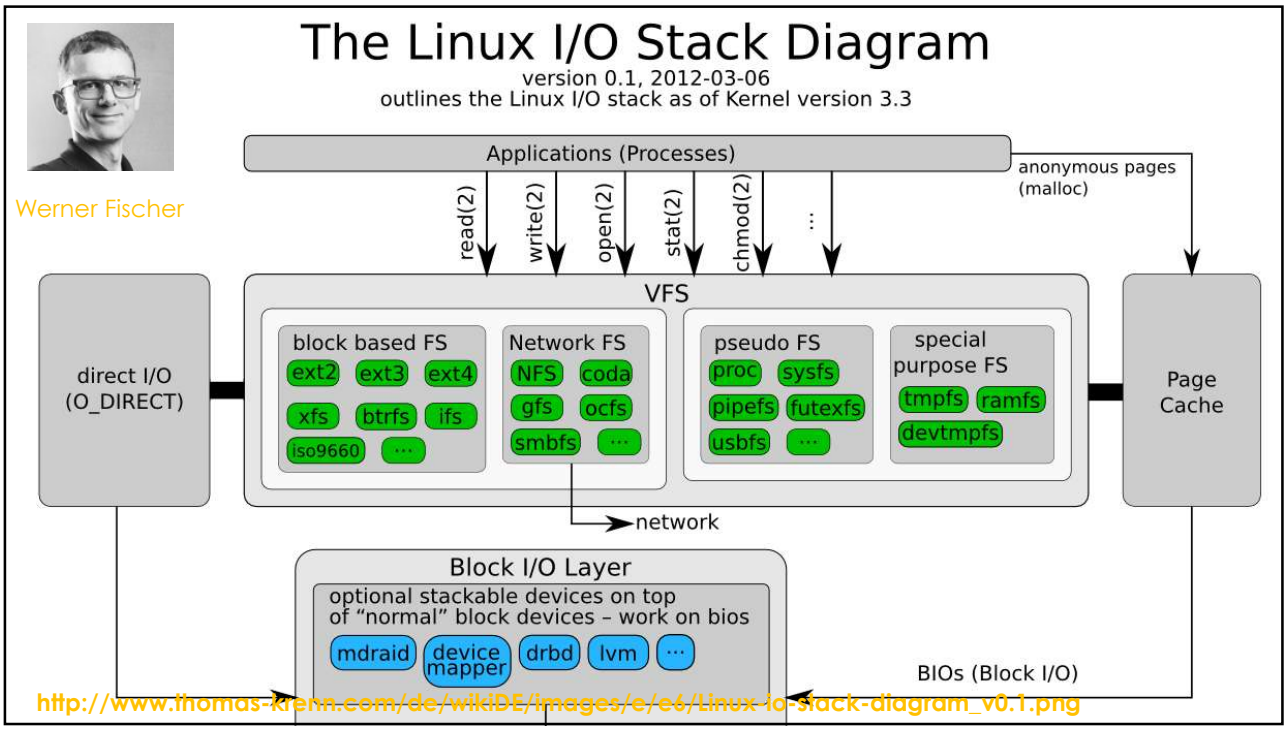


The VFS provides the abstraction layer, separating the POSIX API from the details of how a particular file system implements that behavior.

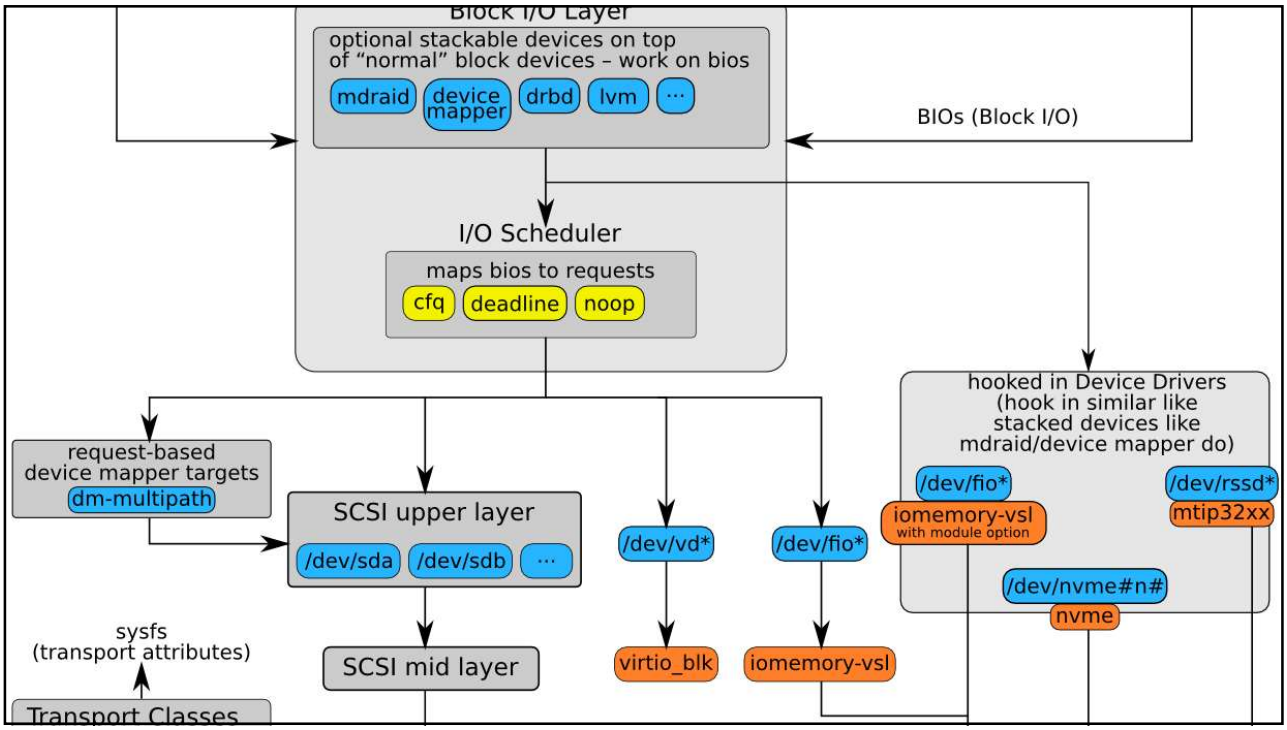
-- Tim Jones

《Linux 虚拟系统文件交换器剖析》

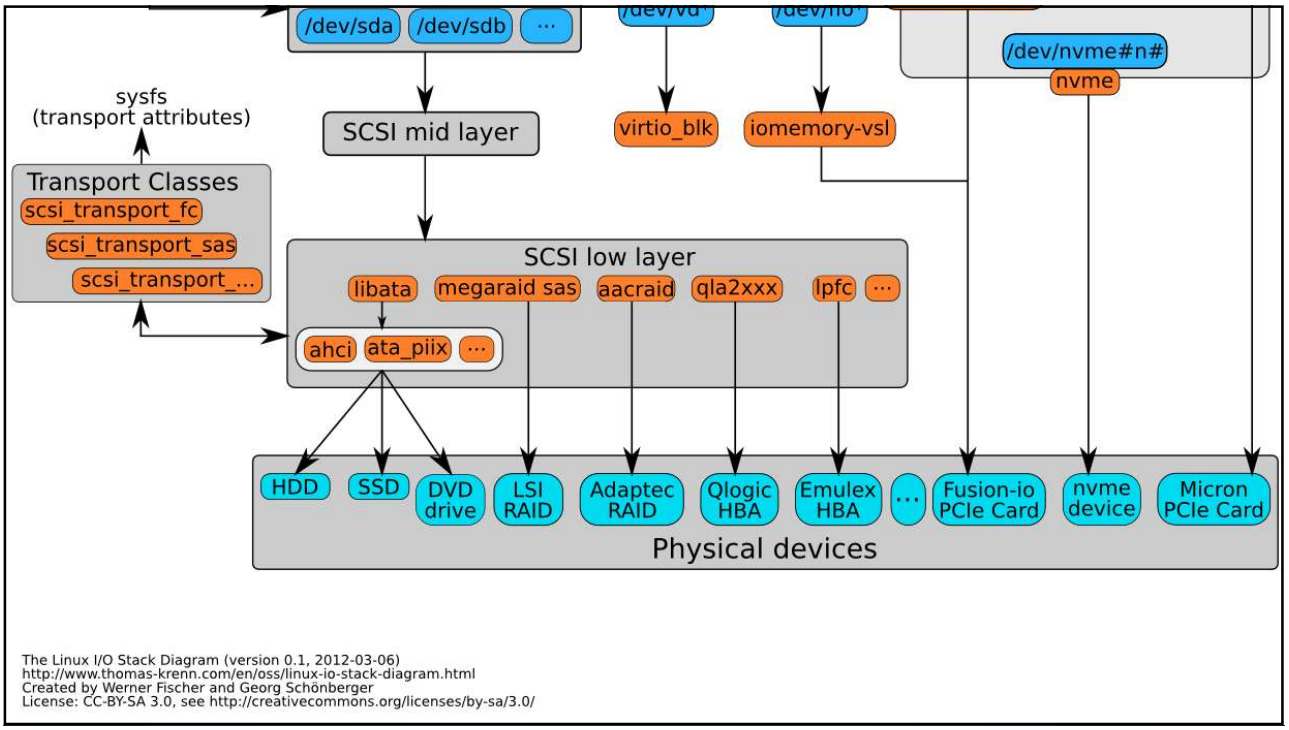
4

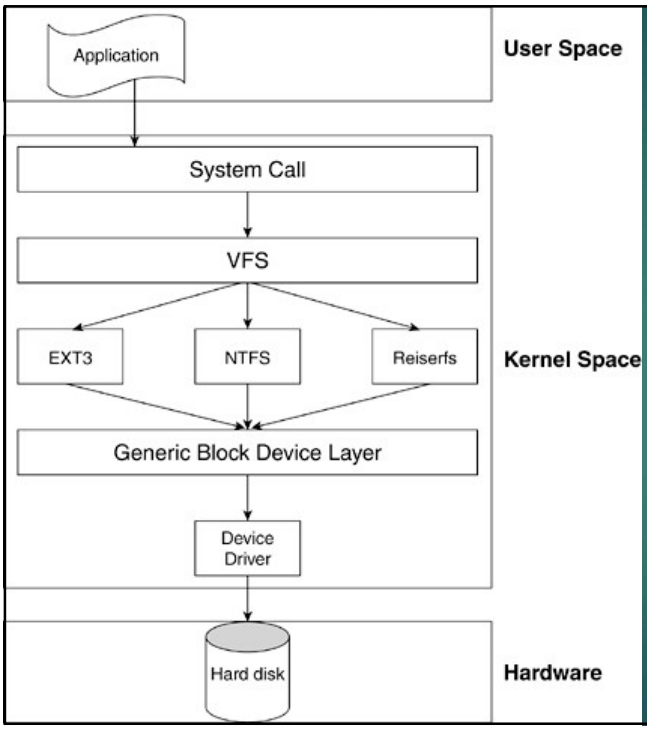

The Linux I/O Stack Diagram
version 0.1, 2012-03-06
outlines the Linux I/O stack as of Kernel version 3.3
Werner Fischer
Applications (Processes)
anonymous pages (malloc)
read(2)
write(2)
open(2)
stat(2)
chmod(2)
...
VFS
direct I/O (O_DIRECT)
block based FS
ext2
ext3
ext4
xfs
btrfs
ifs
iso9660
...
Network FS
NFS
coda
gfs
ocfs
smbfs
...
pseudo FS
proc
sysfs
pipefs
futexfs
usbfs
...
special purpose FS
tmpfs
ramfs
devtmpfs
Page Cache
network
Block I/O Layer
optional stackable devices on top of "normal" block devices – work on bios
mdraid
device mapper
drbd
lvm
...
BIOs (Block I/O)
http://www.thomas-krenn.com/de/wikiDE/images/e/e6/Linux-io-stack-diagram_v0.1.png

5


Block I/O Layer
optional stackable devices on top of "normal" block devices – work on bios
mdraid
device mapper
drbd
lvm
...
BIOs (Block I/O)
I/O Scheduler
maps bios to requests
cfq
deadline
noop
hooked in Device Drivers (hook in similar like stacked devices like mdraid/device mapper do)
/dev/fio*
iomemory-vsl
with module option
/dev/rssd*
mtip32xx
/dev/nvme#n#
nvme
request-based device mapper targets
dm-multipath
SCSI upper layer
/dev/sda
/dev/sdb
...
/dev/vd*
/dev/fio*
sysfs (transport attributes)
SCSI mid layer
virtio_blk
iomemory-vsl
Transport Classes

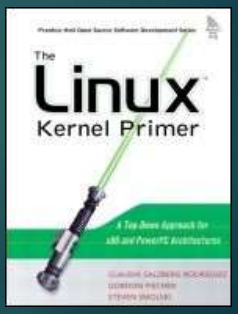
6





Another Diagram

The VFS manages the filesystem-related system calls and translates them to the appropriate filesystem type functions



VFS system calls

open(2)	int open(const char *pathname, int flags);
read(2)	ssize_t read(int fd, void *buf, size_t count);
write(2)	ssize_t write(int fd, const void *buf, size_t count);
lseek(2)	**off_t lseek(int** *fd*, **off_t** *offset*, **int** *whence***);**
fcntl(2)	**int fcntl(int** *fd*, **int** *cmd*, **...** /**arg*/ **);**
close(2)	int close(int fd);
stat(2)	int stat(const char *path, struct stat *buf);
chmod(2)	**int chmod(const char** ***path*, **mode_t** *mode***);**

Filesystem Types

ext2

- The Linux extended filesystem, which was the default for a long time.

ext3

- Essentially ext2 with journaling (logging) to help recover faster after a crash.

ext4

- Performance improvements made on top of ext3.

Pseudo(伪) filesystems

- such as **procfs, sysfs, etc**

/fs/ext2/ext2.h

```
/*
 * Copyright (C) 1992, 1993, 1994, 1995
 * Remy Card (card@masi.ibp.fr)
 * Laboratoire MASI - Institut Blaise Pascal
 * Universite Pierre et Marie Curie (Paris VI)
 *
 *  from
 *
 *  linux/include/linux/minix_fs.h
 *
 *  Copyright (C) 1991, 1992  Linus Torvalds
 */
```

Design and Implementation of the Second Extended Filesystem

Rémy Card, Laboratoire MASI--Institut Blaise Pascal, E-Mail: card@masi.ibp.fr, and
Theodore Ts'o, Massachussets Institute of Technology, E-Mail: tytso@mit.edu, and
Stephen Tweedie, University of Edinburgh, E-Mail: sct@dcs.ed.ac.uk

Stephen Tweedie talking to Linus
LinuxWorld Convention & Expo Winter 99

http://e2fsprogs.sourceforge.net/ext2intro.html

ReiserFS

- ReiserFS is a general-purpose, journaled computer file system formerly designed and implemented by a team at Namesys led by Hans Reiser
- Introduced in version 2.4.1 of the Linux kernel, it was the first journaling file system to be included in the standard kernel
- Hans Thomas Reiser (born December 19, 1963) is an American computer programmer, entrepreneur, and convicted murderer.

file_system_type

```
struct file_system_type {
        const char              *name;      /* filesystem's name */
        int                     fs_flags;   /* filesystem type flags */


        /* the following is used to read the superblock off the disk */
        struct super_block      *(*get_sb) (struct file_system_type *, int,
                                            char *, void *);

        /* the following is used to terminate access to the superblock */
        void                    (*kill_sb) (struct super_block *);

        struct module           *owner;     /* module owning the filesystem */
        struct file_system_type *next;      /* next file_system_type in list */
        struct list_head        fs_supers;  /* list of superblock objects */

        /* the remaining fields are used for runtime lock validation */
        struct lock_class_key   s_lock_key;
        struct lock_class_key   s_umount_key;
        struct lock_class_key   i_lock_key;
        struct lock_class_key   i_mutex_key;
        struct lock_class_key   i_mutex_dir_key;
        struct lock_class_key   i_alloc_sem_key;
};
```

- used to describe a specific variant of a filesystem, such as ext3, ext4, or UDF
- Only one file_system_type per filesystem
- defined in <linux/fs.h>

Register/unregister a filesystem

```
#include <linux/fs.h>

extern int register_filesystem(struct file_system_type *);
extern int unregister_filesystem(struct file_system_type *);
```

- All filesystems registered to the kernel can be seen in the file /proc/filesystems


```
gd@gdbox:~$ cat /proc/filesystems
nodev   sysfs
nodev   rootfs
nodev   bdev
nodev   proc
nodev   cgroup
nodev   cpuset
nodev   tmpfs
nodev   devtmpfs
nodev   debugfs
nodev   securityfs
nodev   sockfs
nodev   pipefs
nodev   anon_inodefs
nodev   devpts
        ext3
        ext4
nodev   ramfs
nodev   hugetlbfs
        vfat
nodev   ecryptfs
        fuseblk
nodev   fuse
nodev   fusectl
nodev   pstore
nodev   mqueue
nodev   vboxsf
        iso9660
```

在KGDB中遍历文件系统

```
(gdb) while $a!=0
 >p *$a
 >set $a=$a->next
 >end
$112 = {name = 0xffffffff81c6f579 "ramfs", fs_flags = 8, mount = 0xffffffff81326e30 <ramfs_m
  next = 0xffffffff81e891e0 <bd_type>, fs_supers = {first = 0x0 <irq_stack_union>}, s_lock_k
  s_writers_key = 0xffffffff81e9a358, i_lock_key = {<No data fields>}, i_mutex_key = {<No da
$113 = {name = 0xffffffff81ca8b17 "bdev", fs_flags = 0, mount = 0xffffffff8126fd20 <bd_mount
  next = 0xffffffff81e8e660 <proc_fs_type>, fs_supers = {first = 0xffff88003e00a8e8}, s_lock
  s_writers_key = 0xffffffff81e89218, i_lock_key = {<No data fields>}, i_mutex_key = {<No da
$114 = {name = 0xffffffff81c9ce5a "proc", fs_flags = 8, mount = 0xffffffff812a7160 <proc_mou
  next = 0xffffffff81e5e540 <cpuset_fs_type>, fs_supers = {first = 0xffff88003e24d8e8}, s_lo
  s_writers_key = 0xffffffff81e8e698, i_lock_key = {<No data fields>}, i_mutex_key = {<No da
$115 = {name = 0xffffffff81ca59d5 "cpuset", fs_flags = 0, mount = 0xffffffff81123210 <cpuset
  next = 0xffffffff81e5ca40 <cgroup_fs_type>, fs_supers = {first = 0x0 <irq_stack_union>}, s
  s_writers_key = 0xffffffff81e5e578, i_lock_key = {<No data fields>}, i_mutex_key = {<No da
$116 = {name = 0xffffffff81ccd46a "cgroup", fs_flags = 8, mount = 0xffffffff8111f8a0 <cgroup
  next = 0xffffffff81e5cca0 <cgroup2_fs_type>, fs_supers = {first = 0xffff88003a9e30e8}, s_l
  s_writers_key = 0xffffffff81e5ca78, i_lock_key = {<No data fields>}, i_mutex_key = {<No da
$117 = {name = 0xffffffff81c9aeb5 "cgroup2", fs_flags = 8, mount = 0xffffffff8111f8a0 <cgrou
  next = 0xffffffff81e6b7c0 <shmem_fs_type>, fs_supers = {first = 0x0 <irq_stack_union>}, s_
  s_writers_key = 0xffffffff81e5ccd8, i_lock_key = {<No data fields>}, i_mutex_key = {<No da
$118 = {name = 0xffffffff81ce2cde "tmpfs", fs_flags = 8, mount = 0xffffffff811bdf50 <shmem_m
  next = 0xffffffff81ed0260 <dev_fs_type>, fs_supers = {first = 0xffff88003abd48e8}, s_lock_
```


挂接(Mount)

- mount [-t vfstype] [-o options] device directory
- mount -t ext4 /dev/mmcblk0p1 /mnt

```
ge@gewubox:~$ cat sf.sh
mount -t vboxsf temp /media/sf_temp
```

- For pseudo filesystem, nodevice can be used
  - mount -t proc nodevice /proc
  - 'mount -t proc proc /proc' also works

VirtualBox的共享文件夹驱动

f:\bench\VirtualBox-4.3.16\src\VBox\Additions\linux\sharedfolders*.*

↑Name	Ext	Size	Date	Attr
[..]		<DIR>	2014-09-10 03:47	----
dirops	c	24,243	2014-09-10 03:47	-a--
files_vboxsf		5,491	2014-09-10 03:47	-a--
lnkops	c	1,791	2014-09-10 03:47	-a--
Makefile	kmk	2,497	2014-09-10 03:47	-a--
Makefile	module	2,749	2014-09-10 03:47	-a--
mount.vboxsf	c	16,468	2014-09-10 03:47	-a--
regops	c	20,046	2014-09-10 03:47	-a--
utils	c	24,686	2014-09-10 03:47	-a--
vbsfmount	c	2,416	2014-09-10 03:47	-a--
vbsfmount	h	2,578	2014-09-10 03:47	-a--
vfsmod	c	16,549	2014-09-10 03:47	-a--
vfsmod	h	5,129	2014-09-10 03:47	-a--

- Labs/sharefilders.zip

/etc/fstab

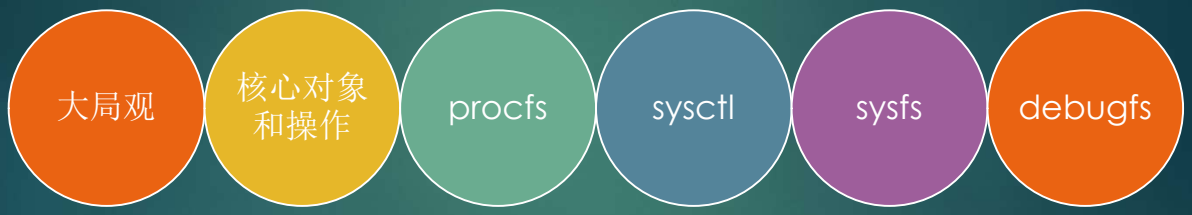
```
ge@gewubox:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    nodev,noexec,nosuid 0       0
# / was on /dev/sda1 during installation
UUID=eb5d8aee-6a0f-4257-b5b8-8d6731ba6764 /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=be9883e4-5e54-4d07-aa82-24ff2e99488e none            swap    sw              0       0
```

vfsmount

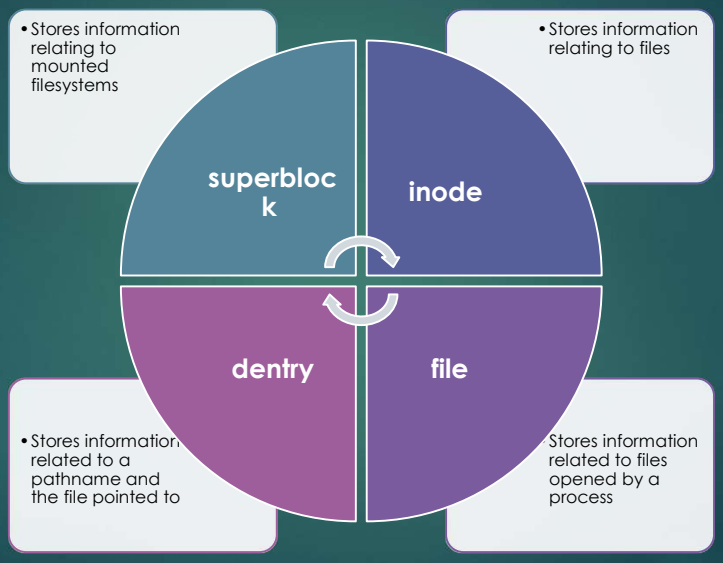
```
struct vfsmount {
        struct list_head   mnt_hash;        /* hash table list */
        struct vfsmount    *mnt_parent;     /* parent filesystem */
        struct dentry      *mnt_mountpoint; /* dentry of this mount point */
        struct dentry      *mnt_root;       /* dentry of root of this fs */
        struct super_block *mnt_sb;         /* superblock of this filesystem */
        struct list_head   mnt_mounts;      /* list of children */
        struct list_head   mnt_child;       /* list of children */
        int                mnt_flags;       /* mount flags */
        char               *mnt_devname;    /* device file name */
        struct list_head   mnt_list;        /* list of descriptors */
        struct list_head   mnt_expire;      /* entry in expiry list */
        struct list_head   mnt_share;       /* entry in shared mounts list */
        struct list_head   mnt_slave_list;  /* list of slave mounts */
        struct list_head   mnt_slave;       /* entry in slave list */
        struct vfsmount    *mnt_master;     /* slave's master */
        struct mnt_namespace *mnt_namespace; /* associated namespace */
        int                mnt_id;          /* mount identifier */
        int                mnt_group_id;    /* peer group identifier */
        atomic_t           mnt_count;       /* usage count */
        int                mnt_expiry_mark; /* is marked for expiration */
        int                mnt_pinned;      /* pinned count */
        int                mnt_ghosts;      /* ghosts count */
        atomic_t           __mnt_writers;   /* writers count */
};
```

$7 = {mnt_root = 0xeec7ee00, mnt_sb = 0xef331c00, mnt_flags = 32}

- represents a specific instance of a filesystem—in other words, a mount point
- defined in <linux/mount.h>



四大对象



四大操作

super_operations	contains the methods that the kernel can invoke on a specific filesystem, such as write_inode() and sync_fs()
inode_operations	contains the methods that the kernel can invoke on a specific file, such as create() and link()
dentry_operations	contains the methods that the kernel can invoke on a specific directory entry, such as d_compare() and d_delete()
file_operations	contains the methods that a process can invoke on an open file, such as read() and write()

superblock

- store information describing that specific filesystem
  - type, size, and status of the mounted filesystem
- corresponds to the *filesystem superblock* or the *filesystem control block*, which is stored in a special sector on disk

dumpe2fs

```
DUMPE2FS(8)                                                          DUMPE2FS(8)

NAME
       dumpe2fs - dump ext2/ext3/ext4 filesystem information

SYNOPSIS
       dumpe2fs [ -bfhixV ] [ -o superblock=superblock ] [ -o blocksize=blocksize ] device

DESCRIPTION
       dumpe2fs  prints  the  super  block  and  blocks  group information for the filesystem present on
       device.

       Note: When used with a mounted filesystem, the printed information may be old or inconsistent.

OPTIONS
       -b     print the blocks which are reserved as bad in the filesystem.

       -o superblock=superblock
              use the block superblock when examining the filesystem.  This option is not usually needed
              except  by  a  filesystem  wizard  who  is examining the remains of a very badly corrupted
              filesystem.

       -o blocksize=blocksize
 Manual page dumpe2fs(8) line 1/60 43% (press h for help or q to quit)
```

```
ge@gewubox:~$ sudo dumpe2fs -h /dev/sda1
dumpe2fs 1.42 (29-Nov-2011)
Filesystem volume name:   <none>
Last mounted on:          /
Filesystem UUID:          eb5d8aee-6a0f-4257-b5b8-8d6731ba6764
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery extent flex_bg
sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              2048000
Block count:              8192000
Reserved block count:     409600
Free blocks:              6716403
Free inodes:              1828087
First block:              0
Block size:               4096
Fragment size:            4096
```

27

```
Reserved GDT blocks:      1022
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8192
Inode blocks per group:   512
Flex block group size:    16
Filesystem created:       Sat Aug 13 20:04:08 2016
Last mount time:          Sun Sep  4 20:31:17 2016
Last write time:          Sat Aug 13 20:22:26 2016
Mount count:              61
Maximum mount count:      -1
Last checked:             Sat Aug 13 20:04:08 2016
Check interval:           0 (<none>)
Lifetime writes:          15 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     28
Desired extra isize:      28
Journal inode:            8
First orphan inode:       1857256
Default directory hash:   half_md4
Directory Hash Seed:      836100b0-a816-4f60-ad14-
ac257e01a7a4
Journal backup:           inode blocks
Journal features:         journal_incompat_revoke
Journal size:             128M
Journal length:           32768
Journal sequence:         0x0000451c
Journal start:            16
```

28

```
struct super_block {
        struct list_head        s_list;           /* list of all superblocks */
        dev_t                   s_dev;            /* identifier */
        unsigned long           s_blocksize;      /* block size in bytes */
        unsigned char           s_blocksize_bits; /* block size in bits */
        unsigned char           s_dirt;           /* dirty flag */
        unsigned long long      s_maxbytes;       /* max file size */
        struct file_system_type s_type;           /* filesystem type */
        struct super_operations s_op;             /* superblock methods */
        struct dquot_operations *dq_op;           /* quota methods */
        struct quotactl_ops     *s_qcop;          /* quota control methods */
        struct export_operations *s_export_op;    /* export methods */
        unsigned long            s_flags;         /* mount flags */
        unsigned long            s_magic;         /* filesystem's magic number */
        struct dentry            *s_root;         /* directory mount point */
        struct rw_semaphore      s_umount;        /* unmount semaphore */
        struct semaphore         s_lock;          /* superblock semaphore */
        int                      s_count;         /* superblock ref count */
        int                      s_need_sync;     /* not-yet-synced flag */
        atomic_t                 s_active;        /* active reference count */
        void                     *s_security;     /* security module */
        struct xattr_handler     **s_xattr;       /* extended attribute handlers */
```

29

```
        struct list_head      s_inodes;        /* list of inodes */
        struct list_head      s_dirty;         /* list of dirty inodes */
        struct list_head      s_io;            /* list of writebacks */
        struct list_head      s_more_io;       /* list of more writeback */
        struct hlist_head     s_anon;          /* anonymous dentries */
        struct list_head      s_files;         /* list of assigned files */
        struct list_head      s_dentry_lru;    /* list of unused dentries */
        int                   s_nr_dentry_unused; /* number of dentries on list */
        struct block_device   *s_bdev;         /* associated block device */
        struct mtd_info       *s_mtd;          /* memory disk information */
        struct list_head      s_instances;     /* instances of this fs */
        struct quota_info     s_dquot;         /* quota-specific options */
        int                   s_frozen;        /* frozen status */
        wait_queue_head_t     s_wait_unfrozen; /* wait queue on freeze */
        char                  s_id[32];        /* text name */
        void                  *s_fs_info;      /* filesystem-specific info */
        fmode_t               s_mode;          /* mount permissions */
        struct semaphore      s_vfs_rename_sem; /* rename semaphore */
        u32                   s_time_gran;     /* granularity of timestamps */
        char                  *s_subtype;      /* subtype name */
        char                  *s_options;      /* saved mount options */
};
```

30

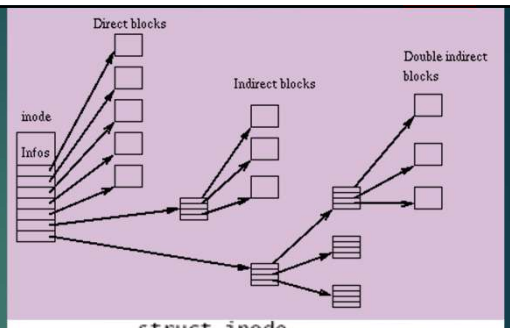
```
$9 = {s_list = {next = 0xef2eb400, prev = 0xd13af400}, s_dev = 8388609,
  s_blocksize_bits = 12 '\f', s_blocksize = 4096, s_maxbytes = 8796093022207,
  s_type = 0xc19876a8, s_op = 0xc16a67e0, dq_op = 0xc16a68c0,
  s_qcop = 0xc16a6920, s_export_op = 0xc16a6898, s_flags = 1879146496,
  s_magic = 61267, s_root = 0xeec7ee00, s_umount = {count = 0, wait_lock = {
      raw_lock = {{head_tail = 0, tickets = {head = 0 '\000',
            tail = 0 '\000'}}}}, wait_list = {next = 0xef331c44,
      prev = 0xef331c44}}, s_count = 1, s_active = {counter = 1},
  s_security = 0x0, s_xattr = 0xc1987de8, s_inodes = {next = 0xe5f0c878,
    prev = 0xeec80128}, s_anon = {first = 0x0}, s_files = 0xc1ac1208,
  s_mounts = {next = 0xee86fdb0, prev = 0xee86fdb0}, s_bdev = 0xeec4bc00,
  s_bdi = 0xef2a00d0, s_mtd = 0x0, s_instances = {next = 0x0,
    pprev = 0xc19876c0}, s_dquot = {flags = 0, dqio_mutex = {count = {
        counter = 1}, wait_lock = {{rlock = {raw_lock = {{head_tail = 0,
                tickets = {head = 0 '\000', tail = 0 '\000'}}}}}},
      wait_list = {next = 0xef331c94, prev = 0xef331c94}, owner = 0x0,
      spin_mlock = 0x0}, dqonoff_mutex = {count = {counter = 1}, wait_lock = {{
          rlock = {raw_lock = {{head_tail = 0, tickets = {head = 0 '\000',
                tail = 0 '\000'}}}}}, wait_list = {next = 0xef331cac,
        prev = 0xef331cac}, owner = 0x0, spin_mlock = 0x0}, dqptr_sem = {
      count = 0, wait_lock = {raw_lock = {{head_tail = 0, tickets = {
            head = 0 '\000', tail = 0 '\000'}}}}, wait_list = {
        next = 0xef331cc4, prev = 0xef331cc4}}, files = {0x0, 0x0}, info = {{
---Type <return> to continue, or q <return> to quit---
      dqi_format = 0x0, dqi_fmt_id = 0, dqi_dirty_list = {next = 0x0,
```


Superblock Operations

```
struct super_operations {
  struct inode *(*alloc_inode)(struct super_block *sb);
  void (*destroy_inode)(struct inode *);
  void (*dirty_inode) (struct inode *);
  int (*write_inode) (struct inode *, int);
  void (*drop_inode) (struct inode *);
  void (*delete_inode) (struct inode *);
  void (*put_super) (struct super_block *);
  void (*write_super) (struct super_block *);
  int (*sync_fs)(struct super_block *sb, int wait);
  int (*freeze_fs) (struct super_block *);
  int (*unfreeze_fs) (struct super_block *);
  int (*statfs) (struct dentry *, struct kstatfs *);
  int (*remount_fs) (struct super_block *, int *, char *);
  void (*clear_inode) (struct inode *);
  void (*umount_begin) (struct super_block *);
  int (*show_options)(struct seq_file *, struct vfsmount *);
  int (*show_stats)(struct seq_file *, struct vfsmount *);
  ssize_t (*quota_read)(struct super_block *, int, char *, size_t, loff_t);
  ssize_t (*quota_write)(struct super_block *, int, const char *, size_t, loff_t);
  int (*bdev_try_to_free_page)(struct super_block*, struct page*, gfp_t);
};
```

inode

- short for *index node*
- 文件的描述信息
  - file type, access rights, owners, timestamps, size, pointers to data blocks.
  - 一个文件需要一个inode
- An inode can refer to a file or a directory or a symbolic link to another object
- An inode represents each file on a filesystem, but the inode object is constructed in memory only as files are accessed



struct inode

```
struct list_head i_dentry;
struct timespec i_atime;
struct timespec i_mtime;
struct timespec i_ctime;
gid_t i_gid;
uid_t i_uid;
loff_t i_size;
const struct file_operations *i_fop;
const struct inode_operations *i_op;
struct address_space *i_mapping;
struct address_space *i_data;
...
```

/linux/include/linux/fs.h


```
struct inode {
        struct hlist_node       i_hash;             /* hash list */
        struct list_head        i_list;             /* list of inodes */
        struct list_head        i_sb_list;          /* list of superblocks */
        struct list_head        i_dentry;           /* list of dentries */
        unsigned long           i_ino;              /* inode number */
        atomic_t                i_count;            /* reference counter */
        unsigned int            i_nlink;            /* number of hard links */
        uid_t                   i_uid;              /* user id of owner */
        gid_t                   i_gid;              /* group id of owner */
        kdev_t                  i_rdev;             /* real device node */
        u64                     i_version;          /* versioning number */
        loff_t                  i_size;             /* file size in bytes */
        seqcount_t              i_size_seqcount;    /* serializer for i_size */
        struct timespec         i_atime;            /* last access time */
        struct timespec         i_mtime;            /* last modify time */
        struct timespec         i_ctime;            /* last change time */
        unsigned int            i_blkbits;          /* block size in bits */
        blkcnt_t                i_blocks;           /* file size in blocks */
        unsigned short          i_bytes;            /* bytes consumed */
        umode_t                 i_mode;             /* access permissions */
```

```
        spinlock_t              i_lock;             /* spinlock */
        struct rw_semaphore     i_alloc_sem;        /* nests inside of i_sem */
        struct semaphore        i_sem;              /* inode semaphore */
        struct inode_operations *i_op;              /* inode ops table */
        struct file_operations  *i_fop;             /* default inode ops */
        struct super_block      *i_sb;              /* associated superblock */
        struct file_lock        *i_flock;           /* file lock list */
        struct address_space    *i_mapping;         /* associated mapping */
        struct address_space    i_data;             /* mapping for device */
        struct dquot            *i_dquot[MAXQUOTAS]; /* disk quotas for inode */
        struct list_head        i_devices;          /* list of block devices */
        union {
            struct pipe_inode_info  *i_pipe;        /* pipe information */
            struct block_device     *i_bdev;        /* block device driver */
            struct cdev             *i_cdev;        /* character device driver */
        };
        unsigned long           i_dnotify_mask;     /* directory notify mask */
        struct dnotify_struct   *i_dnotify;         /* dnotify */
        struct list_head        inotify_watches;    /* inotify watches */
        struct mutex            inotify_mutex;      /* protects inotify_watches */
        unsigned long           i_state;            /* state flags */
        unsigned long           dirtied_when;       /* first dirtying time */
        unsigned int            i_flags;            /* filesystem flags */
        atomic_t                i_writecount;       /* count of writers */
        void                    *i_security;        /* security module */
        void                    *i_private;         /* fs private pointer */
};
```


stat

```
ge@gewubox:~$ stat /boot/vmlinuz-3.12.2
  File: `/boot/vmlinuz-3.12.2'
  Size: 5864384         Blocks: 11456      IO Block: 4096   regular file
Device: 801h/2049d      Inode: 798202      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-08-14 09:16:13.124998872 +0800
Modify: 2016-08-14 09:16:13.160998872 +0800
Change: 2016-08-14 09:16:13.160998872 +0800
 Birth: -
```

df -i

```
ge@gewubox:~$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
/dev/sda1      2048000 220344 1827656   11% /
udev             77839    469   77370    1% /dev
tmpfs            95776    404   95372    1% /run
none             95776      3   95773    1% /run/lock
none             95776      7   95769    1% /run/shm
temp              1000      0    1000    0% /media/sf_temp
/dev/sr0             0      0       0     - /media/VBOXADDITIONS_5.0.26_108824
ge@gewubox:~$ df
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/sda1       32122784   5277900  25190100  18% /
udev              311356         4    311352   1% /dev
tmpfs             153244       796    152448   1% /run
none                5120         0      5120   0% /run/lock
none              383104       200    382904   1% /run/shm
temp           155777020 152769756   3007264  99% /media/sf_temp
/dev/sr0           56868     56868         0 100% /media/VBOXADDITIONS_5.0.26_10
8824
```

- 格式化时确定了总的inode数量，inode用完，那么该文件系统也无法加入新文件

Inode Operations

```
struct inode_operations {
   int (*create) (struct inode *,struct dentry *,int, struct nameidata *);
   struct dentry * (*lookup) (struct inode *,struct dentry *, struct nameidata *);
   int (*link) (struct dentry *,struct inode *,struct dentry *);
   int (*unlink) (struct inode *,struct dentry *);
   int (*symlink) (struct inode *,struct dentry *,const char *);
   int (*mkdir) (struct inode *,struct dentry *,int);
   int (*rmdir) (struct inode *,struct dentry *);
   int (*mknod) (struct inode *,struct dentry *,int,dev_t);
   int (*rename) (struct inode *, struct dentry *,
                  struct inode *, struct dentry *);
   int (*readlink) (struct dentry *, char __user *,int);
   void * (*follow_link) (struct dentry *, struct nameidata *);
   void (*put_link) (struct dentry *, struct nameidata *, void *);
   void (*truncate) (struct inode *);
   int (*permission) (struct inode *, int);
   int (*setattr) (struct dentry *, struct iattr *);
   int (*getattr) (struct vfsmount *mnt, struct dentry *, struct kstat *);
   int (*setxattr) (struct dentry *, const char *,const void *,size_t,int);
   ssize_t (*getxattr) (struct dentry *, const char *, void *, size_t);
   ssize_t (*listxattr) (struct dentry *, char *, size_t);
   int (*removexattr) (struct dentry *, const char *);
   void (*truncate_range)(struct inode *, loff_t, loff_t);
   long (*fallocate)(struct inode *inode, int mode, loff_t offset,
                     loff_t len);
   int (*fiemap)(struct inode *, struct fiemap_extent_info *, u64 start,
                 u64 len);
};
```

Ext4

Lister - [D:\bench\linux-3.16.3\fs\ext4\file.c]

File Edit Options Help

```
const struct inode_operations ext4_file_inode_operations = {
        .setattr        = ext4_setattr,
        .getattr        = ext4_getattr,
        .setxattr       = generic_setxattr,
        .getxattr       = generic_getxattr,
        .listxattr      = ext4_listxattr,
        .removexattr    = generic_removexattr,
        .get_acl        = ext4_get_acl,
        .set_acl        = ext4_set_acl,
        .fiemap         = ext4_fiemap,
};
```



```
(gdb) p *inode
$3 = {i_mode = 33188, i_opflags = 5, i_uid = 0, i_gid = 0, i_flags = 0,
  i_acl = 0x0, i_default_acl = 0xffffffff, i_op = 0xc16a5dc0,
  i_sb = 0xef331c00, i_mapping = 0xeef87264, i_security = 0x0,
  i_ino = 1048787, {i_nlink = 1, __i_nlink = 1}, i_rdev = 0, i_size = 14,
  i_atime = {tv_sec = 1472992280, tv_nsec = 510988000}, i_mtime = {
    tv_sec = 1471090117, tv_nsec = 769871000}, i_ctime = {tv_sec = 1471090117,
    tv_nsec = 769871000}, i_lock = {{rlock = {raw_lock = {{head_tail = 4112,
          tickets = {head = 16 '\020', tail = 16 '\020'}}}}}}, i_bytes = 0,
  i_blkbits = 12, i_blocks = 8, i_size_seqcount = {sequence = 0}, i_state = 0,
  i_mutex = {count = {counter = 1}, wait_lock = {{rlock = {raw_lock = {{
            head_tail = 0, tickets = {head = 0 '\000',
              tail = 0 '\000'}}}}}}, wait_list = {next = 0xeef8720c,
      prev = 0xeef8720c}, owner = 0x0, spin_mlock = 0x0}, dirtied_when = 0,
  i_hash = {next = 0x0, pprev = 0xef933db4}, i_wb_list = {next = 0xeef87228,
    prev = 0xeef87228}, i_lru = {next = 0xeef87230, prev = 0xeef87230},
  i_sb_list = {next = 0xeef86fc8, prev = 0xeef874a8}, {i_dentry = {
      first = 0xeef55478}, i_rcu = {next = 0xeef55478, func = 0}},
  i_version = 1, i_count = {counter = 1}, i_dio_count = {counter = 0},
  i_writecount = {counter = 0}, i_fop = 0xc16a5e40, i_flock = 0x0, i_data = {
    host = 0xeef87198, page_tree = {height = 0, gfp_mask = 32,
      rnode = 0xefbdd0c0}, tree_lock = {{rlock = {raw_lock = {{
            head_tail = 514, tickets = {head = 2 '\002',
              tail = 2 '\002'}}}}}}, i_mmap_writable = 0, i_mmap = {
```

rb_node = 0x0}, i_mmap_nonlinear = {next = 0xeef87280,
prev = 0xeef87280}, i_mmap_mutex = {count = {counter = 1}, wait_lock = {{
rlock = {raw_lock = {{head_tail = 0, tickets = {head = 0 '\000',
tail = 0 '\000'}}}}}}, wait_list = {next = 0xeef87290,
prev = 0xeef87290}, owner = 0x0, spin_mlock = 0x0}, nrpages = 1,
writeback_index = 0, a_ops = 0xc16a6060, flags = 131290,
backing_dev_info = 0xef2a00d0, private_lock = {{rlock = {raw_lock = {{
head_tail = 0, tickets = {head = 0 '\000',
tail = 0 '\000'}}}}}}, private_list = {next = 0xeef872b8,
prev = 0xeef872b8}, private_data = 0x0}, i_dquot = {0x0, 0x0},
i_devices = {next = 0xeef872cc, prev = 0xeef872cc}, {i_pipe = 0x0,
i_bdev = 0x0, i_cdev = 0x0}, i_generation = 1129505636,
i_fsnotify_mask = 0, i_fsnotify_marks = {first = 0x0}, i_private = 0x0}

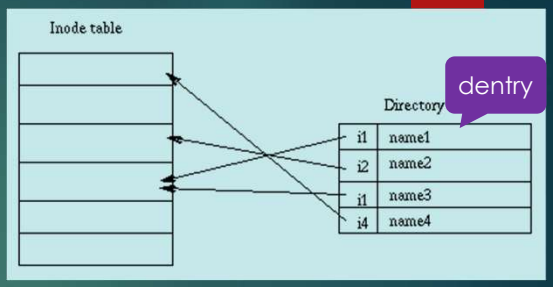
41

```
/dev/sda1: recovering journal
/dev/sda1: Clearing orphaned inode 270550 (uid=0, gid=0, mode=0100644, size=3276
8)
/dev/sda1: Clearing orphaned inode 270166 (uid=0, gid=0, mode=0100600, size=460)
/dev/sda1: Clearing orphaned inode 270167 (uid=0, gid=0, mode=0100644, size=3276
8)
/dev/sda1: Clearing orphaned inode 270165 (uid=0, gid=0, mode=0100600, size=460)
/dev/sda1: Clearing orphaned inode 270163 (uid=0, gid=0, mode=0100644, size=3276
8)
/dev/sda1: Clearing orphaned inode 265675 (uid=0, gid=0, mode=0100600, size=1100
)
/dev/sda1: Clearing orphaned inode 270157 (uid=0, gid=0, mode=0100644, size=4126
)
/dev/sda1: Clearing orphaned inode 266695 (uid=0, gid=0, mode=0100644, size=4126
)
/dev/sda1: Clearing orphaned inode 265189 (uid=0, gid=0, mode=0100644, size=4126
)
/dev/sda1: Clearing orphaned inode 269738 (uid=0, gid=0, mode=0100644, size=4126
)
/dev/sda1: Clearing orphaned inode 264798 (uid=0, gid=0, mode=0100600, size=2621
44)
/dev/sda1: Clearing orphaned inode 1579701 (uid=131, gid=138, mode=0100600, size
=262144)
/dev/sda1: clean, 392939/2031616 files, 3260083/8126208 blocks
_
```

42

dentry

- Directory entry (目录表表项)
- '目录'文件的核心内容
  - the VFS treats directories as a type of file
- Manges the hierarchical nature of a file system
- Only root dentry has not a parent. All other dentries have parents, and some have children
- 有图name1和name3指向同一个inode，其中之一是link



struct dentry

```
struct super_block *d_sb;
struct dentry *d_parent;
struct list_head d_subdirs;
struct dentry_operations *d_op;
unsigned char d_iname[];
struct inod *d_inode;
...
```

/include/linux/dcache.h


```
struct dentry {
        atomic_t                d_count;        /* usage count */
        unsigned int            d_flags;        /* dentry flags */
        spinlock_t              d_lock;         /* per-dentry lock */
        int                     d_mounted;      /* is this a mount point? */
        struct inode            *d_inode;       /* associated inode */
        struct hlist_node       d_hash;         /* list of hash table entries */
        struct dentry           *d_parent;      /* dentry object of parent */
        struct qstr             d_name;         /* dentry name */
        struct list_head        d_lru;          /* unused list */
        union {
            struct list_head    d_child;        /* list of dentries within */
            struct rcu_head     d_rcu;          /* RCU locking */
        } d_u;
        struct list_head        d_subdirs;      /* subdirectories */
        struct list_head        d_alias;        /* list of alias inodes */
        unsigned long           d_time;         /* revalidate time */
        struct dentry_operations *d_op;         /* dentry operations table */
        struct super_block      *d_sb;          /* superblock of file */
        void                    *d_fsdata;      /* filesystem-specific data */
        unsigned char           d_iname[DNAME_INLINE_LEN_MIN]; /* short name */
};
```

filp->f_path.dentry

```
(gdb) p *filp->f_path.dentry
$39 = {d_flags = 128, d_seq = {sequence = 2}, d_hash = {next = 0x0,
   pprev = 0xe5ce5f08}, d_parent = 0xeec23f00, d_name = {{{hash = 2151411773,
      len = 10}, hash_len = 45101084733}, name = 0xeec23c24 "ld-2.15.so"},
  d_inode = 0xeec81dc8,
  d_iname = "ld-
2.15.so\000\000\314\314\314\314\314\314\314\314\314\314\314\314\314\314\314\314\314\
314\314\314\314\314\314", <incomplete sequence \314>,
  d_lockref = {{{lock = {{rlock = {raw_lock = {{head_tail = 13878, tickets = {
                head = 54 '6', tail = 54 '6'}}}}}}, count = 107}}},
  d_op = 0x0, d_sb = 0xef331c00, d_time = 3435973836, d_fsdata = 0x0, d_lru = {
   next = 0xeec23c60, prev = 0xeec23c60}, d_u = {d_child = {
    next = 0xeec23f70, prev = 0xeec231e8}, d_rcu = {next = 0xeec23f70,
    func = 0xeec231e8}}, d_subdirs = {next = 0xeec23c70, prev = 0xeec23c70},
  d_alias = {next = 0x0, pprev = 0xeec81e70}}
```

文件名没有放在inode中，而是放在目录项中，why?


```
#0  ext4_file_open (inode=0xeec81dc8, filp=0xe3435b40) at fs/ext4/file.c:219
#1  0xc1175f7e in do_dentry_open (f=0xe3435b40,
   open=0xc11fb4d0 <ext4_file_open>, cred=0xd90d4f80) at fs/open.c:708
#2  0xc11761f2 in finish_open (file=<optimized out>, dentry=<optimized out>,
   open=0, opened=0xef38fdf0) at fs/open.c:774
#3  0xc1185633 in do_last (nd=0xef38fe14, path=0xef38fddc, file=0xe3435b40,
   op=0xc169d140, opened=0xef38fdf0, name=0xef38fe88) at fs/namei.c:3068
#4  0xc1186055 in path_openat (dfd=<optimized out>, pathname=0xef38fe88,
   nd=0xef38fe14, op=0xc169d140, flags=65) at fs/namei.c:3228
#5  0xc1186645 in do_filp_open (dfd=-100, pathname=0xef38fe88, op=0xc169d140)
   at fs/namei.c:3259
#6  0xc117e4fe in open_exec (name=<optimized out>) at fs/exec.c:766
#7  0xc11bfe0b in load_elf_binary (bprm=0xe3659a00) at fs/binfmt_elf.c:672
#8  0xc117daf9 in search_binary_handler (bprm=<optimized out>)
   at fs/exec.c:1400
#9  search_binary_handler (bprm=0xe3659a00) at fs/exec.c:1374
#10 0xc117ecc8 in exec_binprm (bprm=0xe3659a00) at fs/exec.c:1437
#11 do_execve_common (envp=..., argv=..., filename=<optimized out>)
   at fs/exec.c:1542
#12 do_execve (filename=<optimized out>, __argv=0xb39, __envp=0x9395608)
   at fs/exec.c:1588
#13 0xc117f046 in SYSC_execve (envp=0x9395608, argv=0x939b5c8,
   filename=<optimized out>) at fs/exec.c:1690
```

```
(gdb) p *filp->f_path.dentry.d_parent
$32 = {d_flags = 128, d_seq = {sequence = 2}, d_hash = {next = 0x0,
   pprev = 0xd1e0b108}, d_parent = 0xeec23080, d_name = {{{hash = 4127150537,
      len = 14}, hash_len = 64256692681},
   name = 0xeec23f24 "i386-linux-gnu"}, d_inode = 0xeec81b58,
  d_iname = "i386-linux-
gnu\000\314\314\314\314\314\314\314\314\314\314\314\314\314\314\314\314\314\314\31
4\314", <incomplete sequence \314>, d_lockref = {{{
      lock = {{rlock = {raw_lock = {{head_tail = 64250, tickets = {
                head = 250 '\372', tail = 250 '\372'}}}}}}, count = 71}}},
  d_op = 0x0, d_sb = 0xef331c00, d_time = 3435973836, d_fsdata = 0x0, d_lru = {
   next = 0xeec23f60, prev = 0xeec23f60}, d_u = {d_child = {
     next = 0xeec230f0, prev = 0xe62cd468}, d_rcu = {next = 0xeec230f0,
     func = 0xe62cd468}}, d_subdirs = {next = 0xd7d64d68, prev = 0xeec23c68},
  d_alias = {next = 0x0, pprev = 0xeec81c00}}
(gdb) p *filp->f_path.dentry.d_parent.d_parent
$33 = {d_flags = 128, d_seq = {sequence = 2}, d_hash = {next = 0x0,
   pprev = 0xe5ef2288}, d_parent = 0xeec7ee00, d_name = {{{hash = 6449516,
      len = 3}, hash_len = 12891351404}, name = 0xeec230a4 "lib"},
  d_inode = 0xeec81678,
  d_iname =
"lib\000nalfd]\000\000\314\314\314\314\314\314\314\314\314\314\314\314\314\314\31
4\314\314\314\314\314\314\314", <incomplete sequence \314>,
  d_lockref = {{{lock = {{rlock = {raw_lock = {{head_tail = 22616, tickets = {
```


Dentry Operations

```
struct dentry_operations {
        int (*d_revalidate) (struct dentry *, struct nameidata *);
        int (*d_hash) (struct dentry *, struct qstr *);
        int (*d_compare) (struct dentry *, struct qstr *, struct qstr *);
        int (*d_delete) (struct dentry *);
        void (*d_release) (struct dentry *);
        void (*d_iput) (struct dentry *, struct inode *);
        char *(*d_dname) (struct dentry *, char *, int);
};
```

dcache

- Directory Entry Cache
- Also known as the dentry cache
- Provides a very fast look-up mechanism to translate a pathname (filename) into a specific dentry.
- Dentries live in RAM and are never saved to disc: they exist only for performance.

File Object

- represent a file opened by a process
- the in-memory representation of an open file

struct file

```
struct path f_path;
struct dentry (f_path.dentry);
const struct file_operations *f_op;
unsigned int f_flags;
fmode_t f_mode;
lodd_t f_pos;
...
```

<linux/fs.h>

```
struct file {
        union {
            struct list_head   fu_list;       /* list of file objects */
            struct rcu_head    fu_rcuhead;    /* RCU list after freeing */
        } f_u;
        struct path            f_path;        /* contains the dentry */
        struct file_operations *f_op;         /* file operations table */
        spinlock_t             f_lock;        /* per-file struct lock */
        atomic_t               f_count;       /* file object's usage count */
        unsigned int           f_flags;       /* flags specified on open */
        mode_t                 f_mode;        /* file access mode */
        loff_t                 f_pos;         /* file offset (file pointer) */
        struct fown_struct     f_owner;       /* owner data for signals */
        const struct cred      *f_cred;       /* file credentials */
        struct file_ra_state   f_ra;          /* read-ahead state */
        u64                    f_version;     /* version number */
        void                   *f_security;   /* security module */
        void                   *private_data; /* tty driver hook */
        struct list_head       f_ep_links;    /* list of epoll links */
        spinlock_t             f_ep_lock;     /* epoll lock */
        struct address_space   *f_mapping;    /* page cache mapping */
        unsigned long          f_mnt_write_state; /* debugging state */
};
```



```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*aio_read) (struct kiocb *, const struct iovec *,
                             unsigned long, loff_t);
        ssize_t (*aio_write) (struct kiocb *, const struct iovec *,
                              unsigned long, loff_t);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        int (*ioctl) (struct inode *, struct file *, unsigned int,
                      unsigned long);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *, fl_owner_t id);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, struct dentry *, int datasync);
        int (*aio_fsync) (struct kiocb *, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*sendpage) (struct file *, struct page *,
                             int, size_t, loff_t *, int);
```

```
        unsigned long (*get_unmapped_area) (struct file *,
                                            unsigned long,
                                            unsigned long,
                                            unsigned long,
                                            unsigned long);
        int (*check_flags) (int);
        int (*flock) (struct file *, int, struct file_lock *);
        ssize_t (*splice_write) (struct pipe_inode_info *,
                                 struct file *,
                                 loff_t *,
                                 size_t,
                                 unsigned int);
        ssize_t (*splice_read) (struct file *,
                                loff_t *,
                                struct pipe_inode_info *,
                                size_t,
                                unsigned int);
        int (*setlease) (struct file *, long, struct file_lock **);
};
```

53

debugfs: stat /boot/vmlinuz-3.12.2

```
Inode: 798202   Type: regular    Mode:  0644   Flags: 0x80000
Generation: 833054192    Version: 0x00000000:00000001
User:     0   Group:     0   Size: 5864384
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 11456
Fragment:  Address: 0    Number: 0    Size: 0
 ctime: 0x57afc65d:26629760 -- Sun Aug 14 09:16:13 2016
 atime: 0x57afc65d:1dcd5360 -- Sun Aug 14 09:16:13 2016
 mtime: 0x57afc65d:26629760 -- Sun Aug 14 09:16:13 2016
crtime: 0x57afc65d:1dcd5360 -- Sun Aug 14 09:16:13 2016
Size of extra inode fields: 28
EXTENTS:
(0-1431):3199166-3200597
```

54

Case Study

$5 = {f_u = {fu_list = {next = 0xdd99f600, prev = 0xdd99f600}, fu_llist = {
next = 0xdd99f600}, fu_rcuhead = {next = 0xdd99f600,
func = 0xdd99f600}}, **f_path** = {mnt = 0xee86fd90, dentry = 0xeef55400},
f_inode = 0xeef87198, f_op = 0xc16a5e40, f_lock = {{rlock = {raw_lock = {{
head_tail = 0, tickets = {head = 0 '\000', tail = 0 '\000'}}}}}},
f_sb_list_cpu = 0, f_count = {counter = 1}, f_flags = 32768, f_mode = 29,
f_pos = 0, f_owner = {lock = {raw_lock = {lock = 1048576, write = 1048576}},
pid = 0x0, pid_type = PIDTYPE_PID, uid = 0, euid = 0, signum = 0},
f_cred = 0xd934be00, f_ra = {start = 0, size = 0, async_size = 0,
ra_pages = 0, mmap_miss = 0, prev_pos = 0}, f_version = 0,
f_security = 0xe3690d08, private_data = 0x0, f_ep_links = {
next = 0xdd99f67c, prev = 0xdd99f67c}, f_tfile_llink = {next = 0xdd99f684,
prev = 0xdd99f684}, f_mapping = 0xeef87264}

Ext4

Lister - [D:\bench\linux-3.16.3\fs\ext4\file.c]

File Edit Options Help

```
const struct file_operations ext4_file_operations = {
        .llseek         = ext4_llseek,
        .read           = new_sync_read,
        .write          = new_sync_write,
        .read_iter      = generic_file_read_iter,
        .write_iter     = ext4_file_write_iter,
        .unlocked_ioctl = ext4_ioctl,
#ifdef CONFIG_COMPAT
        .compat_ioctl   = ext4_compat_ioctl,
#endif
        .mmap           = ext4_file_mmap,
        .open           = ext4_file_open,
        .release        = ext4_release_file,
        .fsync          = ext4_sync_file,
        .splice_read    = generic_file_splice_read,
        .splice_write   = iter_file_splice_write,
        .fallocate      = ext4_fallocate,
};
```

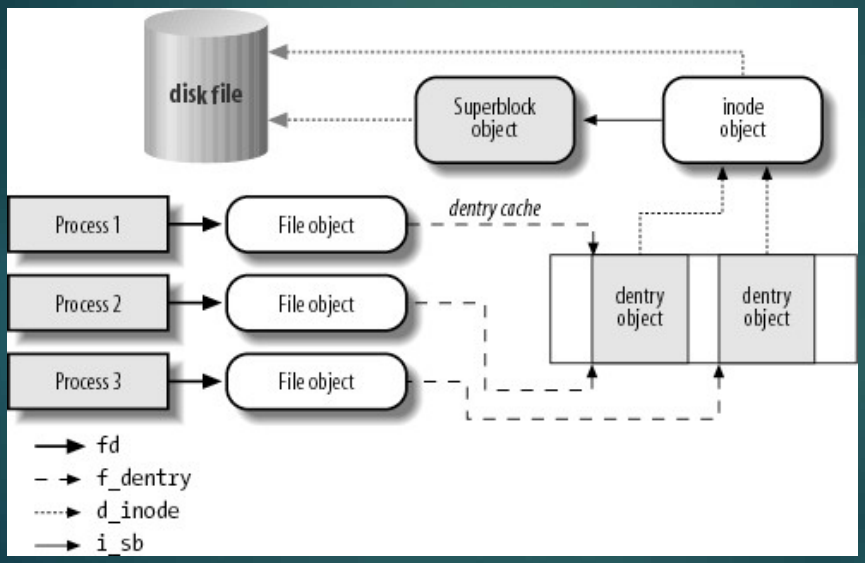
```
#0  ext4_file_open (inode=0xeed50cb8, filp=0xee4cd840) at fs/ext4/file.c:219
#1  0xc1175f7e in do_dentry_open (f=0xee4cd840,
   open=0xc11fb4d0 <ext4_file_open>, cred=0xef371c00) at fs/open.c:708
#2  0xc11761f2 in finish_open (file=<optimized out>, dentry=<optimized out>,
   open=0, opened=0xe644bea8) at fs/open.c:774
#3  0xc1185633 in do_last (nd=0xe644becc, path=0xe644be94, file=0xee4cd840,
   op=0xc169d140, opened=0xe644bea8, name=0xe644bf40) at fs/namei.c:3068
#4  0xc1185d90 in path_openat (dfd=<optimized out>, pathname=0xe644bf40,
   nd=0xe644becc, op=0xc169d140, flags=65) at fs/namei.c:3210
#5  0xc1186645 in do_filp_open (dfd=-100, pathname=0xe644bf40, op=0xc169d140)
   at fs/namei.c:3259
#6  0xc117e4fe in open_exec (name=<optimized out>) at fs/exec.c:766
#7  0xc117e9df in do_execve_common (envp=..., argv=...,
   filename=0xef002010 "/sbin/init") at fs/exec.c:1502
#8  do_execve (filename=0xef002010 "/sbin/init", __argv=0xbf9d92cc,
   __envp=0xbf9d92d8) at fs/exec.c:1588
#9  0xc117f046 in SYSC_execve (envp=0xbf9d92d8, argv=0xbf9d92cc,
   filename=<optimized out>) at fs/exec.c:1690
#10 SyS_execve (filename=-1080189188, argv=-1080192308, envp=-1080192296)
   at fs/exec.c:1685
#11 <signal handler called>
#12 0x0600b90a in ?? ()
```


三个ioctl

- methods.unlocked_ioctl() is the same as ioctl(), except it is called without the Big Kernel Lock (BKL)
- compat_ioctl() is also called without the BKL, but its purpose is to provide a 32-bit compatible ioctl method for 64-bit systems

```
int (*ioctl) (struct inode *, struct file *, unsigned int,
              unsigned long);
long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
```

交互



Case study

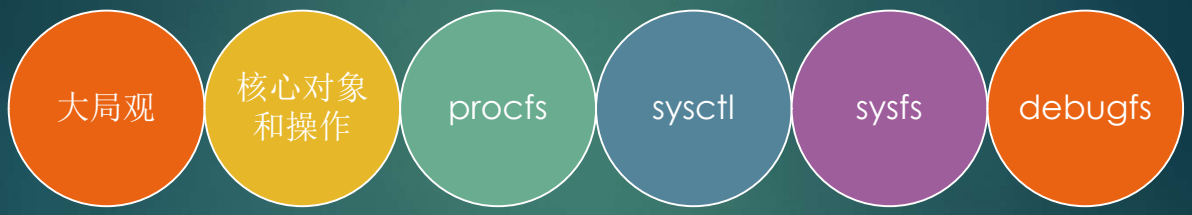
Mar 8 16:31:27 localhost kernel: EXT4-fs error (device dm-12): ext4_mb_generate_buddy:757: group 385, block bitmap and bg descriptor inconsistent: 27138 vs 27163 free clusters

- Mb – multi-block
- super_block *sb
- Bg – block group

Block Group

- An ext4 file system is split into a series of block groups. To reduce performance difficulties due to fragmentation, the block allocator tries very hard to keep each file's blocks within the same group, thereby reducing seek times. The size of a block group is specified in sb.s_blocks_per_group blocks, though it can also calculated as 8 * block_size_in_bytes. With the default block size of 4KiB, each group will contain 32,768 blocks, for a length of 128MiB. The number of block groups is the size of the device divided by the size of a block group.

挂接冲突



63

What it's

- a "pseudo" or "virtual" file system, non-file data represented as a hierarchical file system that doesn't actually exist on disk
- originally designed to allow access to process information, but has since grown to encompass other kernel and in-memory data
  - /proc/cpuinfo
  - /proc/interrupts
- originally implemented in Version 8 UNIX
- allows userland utilities to access information that would normally be restricted to kernel space

64

```
gd@gdbox:/proc$ ls
1     1673  19    2246  30   716        diskstats      pagetypeinfo
10    1676  1968  2252  31   755        dma            partitions
1000  1677  1977  2258  321  778        driver         sched_debug
1080  1684  2     227   324  8          execdomains    schedstat
1082  1696  20    2271  43   822        fb             scsi
11    1698  2084  228   441  829        filesystems    self
1124  17    2086  229   45   839        fs             slabinfo
1152  1704  2089  23    450  841        interrupts     softirqs
12    1706  2090  230   46   843        iomem          stat
13    1717  2092  2327  465  865        ioports        swaps
1376  1719  2097  2330  47   867        irq            sys
1389  1722  21    2337  48   868        kallsyms       sysrq-trigger
1399  1723  210   2342  5    870        kcore          sysvipc
14    1726  2124  2348  50   871        key-users      timer_list
1490  1731  2132  2372  503  891        kmsg           timer_stats
15    1734  2134  2374  504  9          kpagecount     tty
1591  1736  2152  2393  517  956        kpageflags     uptime
16    1739  2167  2405  539  acpi       latency_stats  version
1602  1740  2169  2415  548  asound     loadavg        version_signature
1641  1746  2172  2421  572  buddyinfo  locks          vmallocinfo
1642  1779  2173  2498  614  bus        mdstat         vmstat
1652  1781  2199  25    615  cgroups    meminfo        zoneinfo
1653  1782  22    26    693  cmdline    misc
1663  1792  2205  27    7    consoles   modules
1664  1797  2207  28    70   cpuinfo    mounts
1669  18    2213  29    71   crypto     mtrr
1670  1899  2230  3     712  devices    net
```


Process specific entries in /proc

```
File          Content
clear_refs    Clears page referenced bits shown in smaps output
cmdline       Command line arguments
cpu           Current and last cpu in which it was executed   (2.4)(smp)
cwd           Link to the current working directory
environ       Values of environment variables
exe           Link to the executable of this process
fd            Directory, which contains all file descriptors
maps          Memory maps to executables and library files    (2.4)
mem           Memory held by this process
root          Link to the root directory of this process
stat          Process status
statm         Process memory status information
status        Process status in human readable form
wchan         If CONFIG_KALLSYMS is set, a pre-decoded wchan
pagemap       Page table
stack         Report full stack trace, enable via CONFIG_STACKTRACE
smaps         a extension based on maps, showing the memory consumption of
              each mapping and flags associated with it
```

>cat /proc/self/status

```
gd@gdbox:/proc/sys/fs$ cat /proc/self/status
Name:   cat
State:  R (running)
Tgid:   2520
Pid: 2520
PPid:   2421
TracerPid:  0
Uid: 1000   1000   1000   1000
Gid:   1000   1000   1000   1000
FDSize:  256
Groups: 4 20 24 27 30 46 109 124 1000
VmPeak:     4228 kB
VmSize:   4228 kB
VmLck:     0 kB
VmPin:     0 kB
VmHWM:     280 kB
VmRSS:   280 kB
VmData:     160 kB
VmStk:   136 kB
VmExe:   44 kB
VmLib:   1808 kB
VmPTE:   24 kB
VmSwap:     0 kB
Threads:   1
SigQ:   0/5843
SigPnd: 0000000000000000
ShdPnd:0000000000000000
SigBlk:  0000000000000000
SigIgn:  0000000000000000
SigCgt:  0000000000000000
CapInh:0000000000000000
CapPrm:   0000000000000000
CapEff: 0000000000000000
CapBnd:   0000001ffffffff
Seccomp:  0
Cpus_allowed:  1
Cpus_allowed_list:  0
Mems_allowed: 1
Mems_allowed_list:  0
voluntary_ctxt_switches: 0
nonvoluntary_ctxt_switches: 0
```



```
gd@gdbox:/proc/324$ cat status
Name:  udevd
State:  S (sleeping)
Tgid:   324
Pid: 324
PPid:   1
TracerPid:  0
Uid: 0   0   0   0
Gid:   0   0   0   0
FDSize:  32
Groups:
VmPeak:     3388 kB
VmSize:   3256 kB
VmLck:     0 kB
VmPin:     0 kB
VmHWM:     1516 kB
VmRSS:   176 kB
VmData:     600 kB
VmStk:   136 kB
VmExe:   172 kB
VmLib:   2256 kB
0
VmPTE:   20 kB
VmSwap:     728 kB
Threads:   1
SigQ:   0/5843
SigPnd: 0000000000000000
ShdPnd:0000000000000000
SigBlk:  fffffffe7ffbfeff
SigIgn:  0000000000000000
SigCgt:  0000000180000000
CapInh:0000000000000000
CapPrm:   0000001ffffffff
CapEff: 0000001ffffffff
CapBnd:   0000001ffffffff
Seccomp:  0
Cpus_allowed:  1
Cpus_allowed_list:  0
Mems_allowed: 1
Mems_allowed_list:  0
voluntary_ctxt_switches: 169
nonvoluntary_ctxt_switches: 34
```

/proc/PID/maps

- currently mapped memory regions

```
b73fa000-b743d000 rw-p 00000000 00:00 0
b748c000-b7497000 r-xp 00000000 08:01 656412    /lib/i386-linux-gnu/libnss_files-2.15.so
b7497000-b7498000 r--p 0000a000 08:01 656412    /lib/i386-linux-gnu/libnss_files-2.15.so
b7498000-b7499000 rw-p 0000b000 08:01 656412    /lib/i386-linux-gnu/libnss_files-2.15.so
b7499000-b74a3000 r-xp 00000000 08:01 656416    /lib/i386-linux-gnu/libnss_nis-2.15.so
b74a3000-b74a4000 r--p 00009000 08:01 656416    /lib/i386-linux-gnu/libnss_nis-2.15.so
b74a4000-b74a5000 rw-p 0000a000 08:01 656416    /lib/i386-linux-gnu/libnss_nis-2.15.so
b74a5000-b74bb000 r-xp 00000000 08:01 656406    /lib/i386-linux-gnu/libnsl-2.15.so
b74bb000-b74bc000 r--p 00015000 08:01 656406    /lib/i386-linux-gnu/libnsl-2.15.so
b74bc000-b74bd000 rw-p 00016000 08:01 656406    /lib/i386-linux-gnu/libnsl-2.15.so
b74bd000-b74bf000 rw-p 00000000 00:00 0
b74cf000-b74d1000 rw-p 00000000 00:00 0
b74d1000-b74e8000 r-xp 00000000 08:01 656441    /lib/i386-linux-gnu/libpthread-2.15.so
b74e8000-b74e9000 r--p 00016000 08:01 656441    /lib/i386-linux-gnu/libpthread-2.15.so
b74e9000-b74ea000 rw-p 00017000 08:01 656441    /lib/i386-linux-gnu/libpthread-2.15.so
b74ea000-b74ec000 rw-p 00000000 00:00 0
```



```
b74ec000-b74ef000 r-xp 00000000 08:01 656374    /lib/i386-linux-gnu/libdl-2.15.so
b74ef000-b74f0000 r--p 00002000 08:01 656374    /lib/i386-linux-gnu/libdl-2.15.so
b74f0000-b74f1000 rw-p 00003000 08:01 656374    /lib/i386-linux-gnu/libdl-2.15.so
b74f1000-b7695000 r-xp 00000000 08:01 656361    /lib/i386-linux-gnu/libc-2.15.so
b7695000-b7697000 r--p 001a4000 08:01 656361    /lib/i386-linux-gnu/libc-2.15.so
b7697000-b7698000 rw-p 001a6000 08:01 656361    /lib/i386-linux-gnu/libc-2.15.so
b7698000-b769c000 rw-p 00000000 00:00 0
b769c000-b76a3000 r-xp 00000000 08:01 656447    /lib/i386-linux-gnu/librt-2.15.so
b76a3000-b76a4000 r--p 00006000 08:01 656447    /lib/i386-linux-gnu/librt-2.15.so
b76a4000-b76a5000 rw-p 00007000 08:01 656447    /lib/i386-linux-gnu/librt-2.15.so
b76a5000-b76c2000 r-xp 00000000 08:01 656449    /lib/i386-linux-gnu/libselinux.so.1
b76c2000-b76c3000 r--p 0001c000 08:01 656449    /lib/i386-linux-gnu/libselinux.so.1
b76c3000-b76c4000 rw-p 0001d000 08:01 656449    /lib/i386-linux-gnu/libselinux.so.1
b76c9000-b76d0000 r-xp 00000000 08:01 656408    /lib/i386-linux-gnu/libnss_compat-2.15.so
b76d0000-b76d1000 r--p 00006000 08:01 656408    /lib/i386-linux-gnu/libnss_compat-2.15.so
b76d1000-b76d2000 rw-p 00007000 08:01 656408    /lib/i386-linux-gnu/libnss_compat-2.15.so
b76d3000-b76d4000 rw-p 00000000 00:00 0
b76d4000-b76d6000 rw-p 00000000 00:00 0
b76d6000-b76d7000 r-xp 00000000 00:00 0         [vdso]
b76d7000-b76f7000 r-xp 00000000 08:01 656341    /lib/i386-linux-gnu/ld-2.15.so
b76f7000-b76f8000 r--p 0001f000 08:01 656341    /lib/i386-linux-gnu/ld-2.15.so
b76f8000-b76f9000 rw-p 00020000 08:01 656341    /lib/i386-linux-gnu/ld-2.15.so
b76f9000-b7724000 r-xp 00000000 08:01 393390    /sbin/udevd
b7724000-b7725000 r--p 0002a000 08:01 393390    /sbin/udevd
b7725000-b7726000 rw-p 0002b000 08:01 393390    /sbin/udevd
b861d000-b863e000 rw-p 00000000 00:00 0         [heap]
b863e000-b8662000 rw-p 00000000 00:00 0         [heap]
bff0e000-bff2f000 rw-p 00000000 00:00 0         [stack]
```

> cat /proc/interrupts

```
 CPU0
  0:        36   XT-PIC-XT-PIC    timer
  1:      3596   XT-PIC-XT-PIC    i8042
  2:         0   XT-PIC-XT-PIC    cascade
  4:     10126   XT-PIC-XT-PIC    serial
  8:         0   XT-PIC-XT-PIC    rtc0
  9:     28455   XT-PIC-XT-PIC    acpi, vboxguest
 10:      3043   XT-PIC-XT-PIC    eth0
 11:     26184   XT-PIC-XT-PIC    ohci_hcd:usb1, ahci,
snd_intel8x0
 12:      2536   XT-PIC-XT-PIC    i8042
 14:         0   XT-PIC-XT-PIC    ata_piix
 15:      5517   XT-PIC-XT-PIC    ata_piix
NMI:         0   Non-maskable interrupts
LOC:    318146   Local timer interrupts
SPU:         0   Spurious interrupts
PMI:         0   Performance monitoring interrupts
IWI:      3747   IRQ work interrupts
RTR:         0   APIC ICR read retries
RES:         0   Rescheduling interrupts
CAL:         0   Function call interrupts
TLB:         0   TLB shootdowns
TRM:         0   Thermal event interrupts
THR:         0   Threshold APIC interrupts
MCE:         0   Machine check exceptions
MCP:        19   Machine check polls
ERR:         0
MIS:         0
```

71

> cat /proc/meminfo

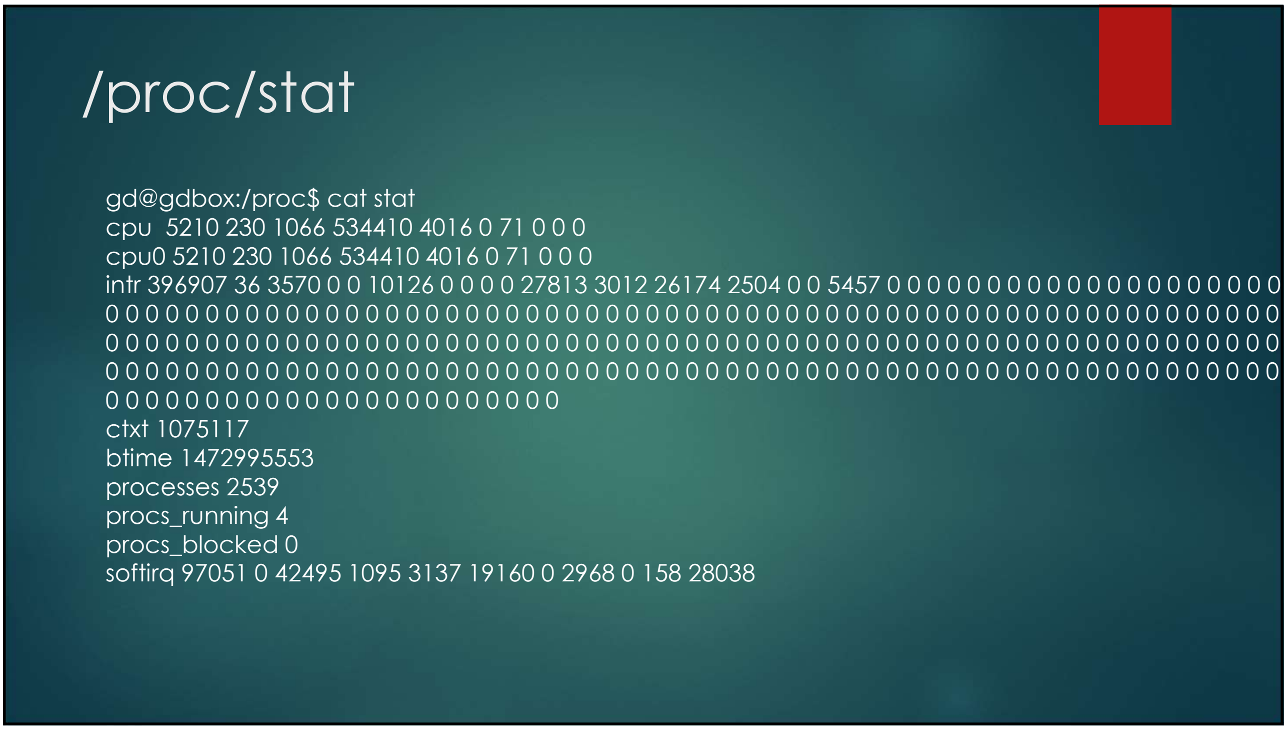
```
MemTotal:        765608 kB
MemFree:          66644 kB
Buffers:           5416 kB
Cached:           41752 kB
SwapCached:       99952 kB
Active:          323104 kB
Inactive:        321228 kB
Active(anon):    297424 kB
Inactive(anon):  302336 kB
Active(file):     25680 kB
Inactive(file):   18892 kB
Unevictable:          0 kB
Mlocked:              0 kB
HighTotal:            0 kB
HighFree:             0 kB
LowTotal:        765608 kB
LowFree:          66644 kB
SwapTotal:       783356 kB
SwapFree:        433368 kB
Dirty:                0 kB
Writeback:            0 kB
AnonPages:       549140 kB
Mapped:           33484 kB
Shmem:             2596 kB
Slab:             23340 kB
SReclaimable:     12512 kB
SUnreclaim:       10828 kB
KernelStack:       2632 kB
PageTables:        7316 kB
NFS_Unstable:         0 kB
Bounce:               0 kB
WritebackTmp:         0 kB
CommitLimit:    1166160 kB
Committed_AS:   2538032 kB
VmallocTotal:    249912 kB
VmallocUsed:      20824 kB
VmallocChunk:    227636 kB
HardwareCorrupted:    0 kB
AnonHugePages:        0 kB
HugePages_Total:      0
HugePages_Free:       0
HugePages_Rsvd:       0
HugePages_Surp:       0
Hugepagesize:      2048 kB
DirectMap4k:      40896 kB
DirectMap2M:     745472 kB
```

72

```
gd@gdbox:/proc$ sudo cat vmstat
nr_free_pages 15516
nr_inactive_anon 75899
nr_active_anon 74669
nr_inactive_file 4162
nr_active_file 7528
nr_unevictable 0
nr_mlock 0
nr_anon_pages 137988
nr_mapped 8536
nr_file_pages 37238
nr_dirty 6
nr_writeback 0
nr_slab_reclaimable 3131
nr_slab_unreclaimable 2733
nr_page_table_pages 1842
nr_kernel_stack 329
nr_unstable 0
nr_bounce 0
nr_vmscan_write 92721
nr_vmscan_immediate_reclaim 69
nr_writeback_temp 0
nr_isolated_anon 0
nr_isolated_file 0
nr_shmem 676
nr_dirtied 3952
nr_written 94151
nr_anon_transparent_hugepages 0
nr_free_cma 0
nr_dirty_threshold 15331
nr_dirty_background_threshold 7665
pgpgin 613185
pgpgout 385572
pswpin 26143
pswpout 92721
pgalloc_dma 18938
pgalloc_normal 1820654
pgalloc_high 0
pgalloc_movable 0
pgfree 1855165
pgactivate 127480
pgdeactivate 257131
pgfault 2110510
pgmajfault 7811
pgrefill_dma 2869
pgrefill_normal 280423
pgrefill_high 0
pgrefill_movable 0
```

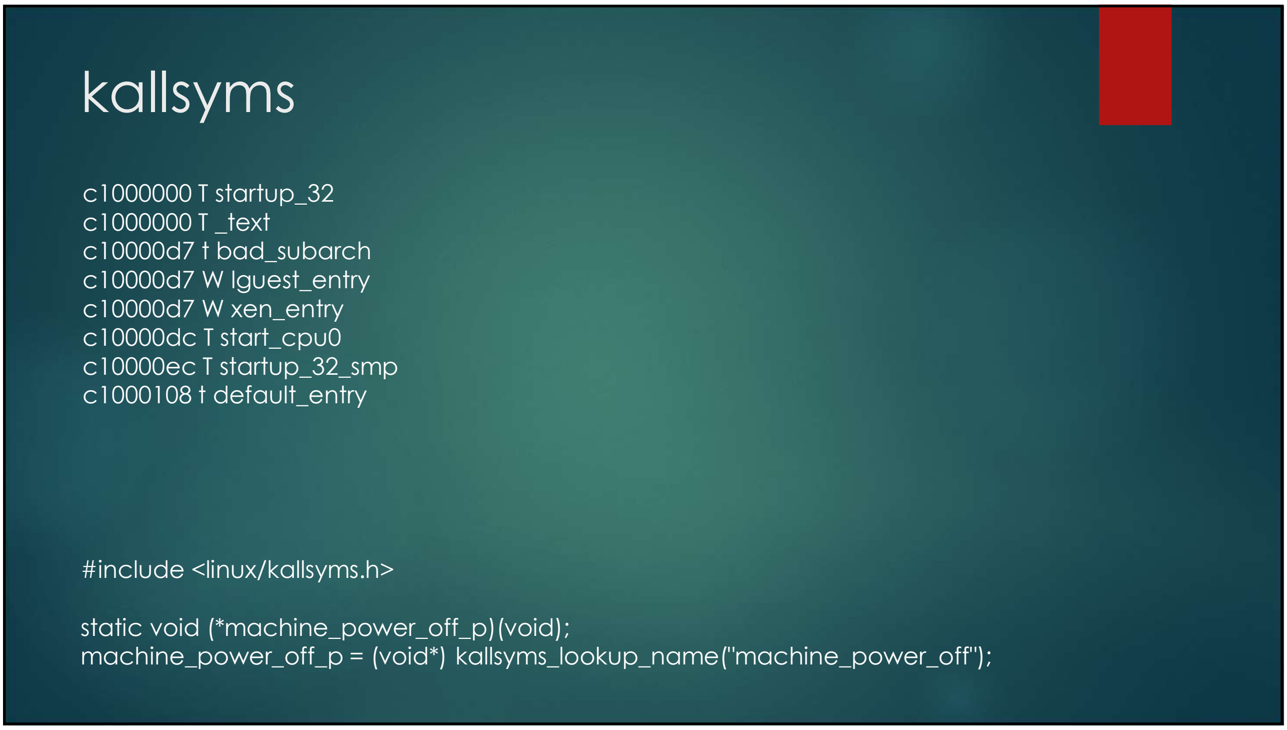


```
pgsteal_kswapd_dma 202
pgsteal_kswapd_normal 198776
pgsteal_kswapd_high 0
pgsteal_kswapd_movable 0
pgsteal_direct_dma 0
pgsteal_direct_normal 3641
pgsteal_direct_high 0
pgsteal_direct_movable 0
pgscan_kswapd_dma 1587
pgscan_kswapd_normal 397225
pgscan_kswapd_high 0
pgscan_kswapd_movable 0
pgscan_direct_dma 0
pgscan_direct_normal 19293
pgscan_direct_high 0
pgscan_direct_movable 0
pgscan_direct_throttle 0
pginodesteal 0
slabs_scanned 58496
kswapd_inodesteal 365
kswapd_low_wmark_hit_quickly 2
kswapd_high_wmark_hit_quickly 51
pageoutrun 82
allocstall 78
pgrotated 92737
pgmigrate_success 0
pgmigrate_fail 0
compact_migrate_scanned 0
compact_free_scanned 0
compact_isolated 0
compact_stall 1
compact_fail 0
compact_success 1
htlb_buddy_alloc_success 0
htlb_buddy_alloc_fail 0
unevictable_pgs_culled 0
unevictable_pgs_scanned 0
unevictable_pgs_rescued 10
unevictable_pgs_mlocked 10
unevictable_pgs_munlocked 10
unevictable_pgs_cleared 0
unevictable_pgs_stranded 0
thp_fault_alloc 0
thp_fault_fallback 0
thp_collapse_alloc 1
thp_collapse_alloc_failed 0
thp_split 0
thp_zero_page_alloc 0
thp_zero_page_alloc_failed 0
```



/proc/stat

```
gd@gdbox:/proc$ cat stat
cpu  5210 230 1066 534410 4016 0 71 0 0 0
cpu0 5210 230 1066 534410 4016 0 71 0 0 0
intr 396907 36 3570 0 0 10126 0 0 0 0 27813 3012 26174 2504 0 0 5457 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ctxt 1075117
btime 1472995553
processes 2539
procs_running 4
procs_blocked 0
softirq 97051 0 42495 1095 3137 19160 0 2968 0 158 28038
```



kallsyms

```
c1000000 T startup_32
c1000000 T _text
c10000d7 t bad_subarch
c10000d7 W lguest_entry
c10000d7 W xen_entry
c10000dc T start_cpu0
c10000ec T startup_32_smp
c1000108 t default_entry
```

```
#include <linux/kallsyms.h>

static void (*machine_power_off_p)(void);
machine_power_off_p = (void*) kallsyms_lookup_name("machine_power_off");
```

Token/Type

- GCC编译时产生（指定）
- # man nm中有详细描述
- 大写代表导出

extern const u8 kallsyms_names[] __weak; extern const u8 kallsyms_token_table[] __weak;extern const u16 kallsyms_token_index[] __weak;

```
/*
 * Label it "global" if it is exported,
 * "local" if not exported.
 */
type = iter->exported ? toupper(iter->type)
                      tolower(iter->type);
```

```
/*
 *
 *
 */
stn
/*
 *
 *
 */
ret
}
```

字符	含义
A	符号值是绝对地址
B/b	位于未初始化数据区（BSS）
C	普通
D/d	位于初始化数据区
G/g	初始化数据区的小对象
i	对于PE，针对特定DLL实现，对于ELF，间接函数
I	对另一个符号的间接引用
N	调试符号
p	位于栈展开区
S/s	未初始化数据区的小对象
T/t	位于代码区
U	未定义
u	唯一的全局符号
V/v	弱对象
W/w	弱符号，声明时指定.weak属性
-	Stabs（ELF中的符号表）符号
?	未知符号


```
gd@gdbox:/proc$ cat cpuinfo
processor    : 0
vendor_id    : GenuineIntel
cpu family   : 6
model        : 60
model name      : Intel(R) Core(TM) i5-4300M CPU @ 2.60GHz
stepping     : 3
cpu MHz          : 2593.994
cache size   : 3072 KB
fdiv_bug     : no
f00f_bug     : no
coma_bug  : no
fpu       : yes
fpu_exception   : yes
cpuid level  : 13
wp        : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fx
nx rdtscp constant_tsc xtopology nonstop_tsc pni pclmulqdq monitor ssse3 cx16 sse4_1 sse4_2 movbe pop
xsave avx rdrand hypervisor lahf_lm abm
bogomips    : 5187.98
clflush size    : 64
cache_alignment     : 64
address sizes      : 39 bits physical, 48 bits virtual
power management:
```

```
gd@gdbox:/proc$ cat ioports
0000-001f : dma1
0020-0021 : pic1
0040-0043 : timer0
0050-0053 : timer1
0060-0060 : keyboard
0064-0064 : keyboard
0070-0071 : rtc_cmos
  0070-0071 : rtc0
0080-008f : dma page reg
00a0-00a1 : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : 0000:00:01.1
  0170-0177 : ata_piix
01f0-01f7 : 0000:00:01.1
  01f0-01f7 : ata_piix
0376-0376 : 0000:00:01.1
  0376-0376 : ata_piix
03c0-03df : vesafb
03f6-03f6 : 0000:00:01.1
  03f6-03f6 : ata_piix
03f8-03ff : serial
0cf8-0cff : PCI conf1
4000-4003 : ACPI PM1a_EVT_BLK
4004-4005 : ACPI PM1a_CNT_BLK
4008-400b : ACPI PM_TMR
4020-4021 : ACPI GPE0_BLK
d000-d00f : 0000:00:01.1
  d000-d00f : ata_piix
d010-d017 : 0000:00:03.0
  d010-d017 : e1000
d020-d03f : 0000:00:04.0
d100-d1ff : 0000:00:05.0
  d100-d1ff : Intel 82801AA-ICH
d200-d23f : 0000:00:05.0
  d200-d23f : Intel 82801AA-ICH
d240-d247 : 0000:00:0d.0
  d240-d247 : ahci
d250-d257 : 0000:00:0d.0
  d250-d257 : ahci
d260-d26f : 0000:00:0d.0
  d260-d26f : ahci
```



```
gd@gdbox:/proc$ sudo cat modules
nls_utf8 12493 1 - Live 0xf0a05000
isofs 39589 1 - Live 0xf1b19000
vesafb 13540 1 - Live 0xf0a0a000 (F)
crc32_pclmul 13006 0 - Live 0xf09fb000
vboxsf 38472 2 - Live 0xf1b02000 (OF)
snd_intel8x0 33458 4 - Live 0xf0a57000
snd_ac97_codec 110295 1 snd_intel8x0, Live 0xf0ac5000
ac97_bus 12642 1 snd_ac97_codec, Live 0xf09e6000
snd_pcm 94597 3 snd_intel8x0,snd_ac97_codec, Live 0xf0ae4000
snd_seq_midi 13132 0 - Live 0xf0a00000
aesni_intel 18196 0 - Live 0xf09ec000
snd_rawmidi 25157 1 snd_seq_midi, Live 0xf09f3000
ablk_helper 13357 1 aesni_intel, Live 0xf0958000
cryptd 19821 1 ablk_helper, Live 0xf09e0000
lrw 13127 1 aesni_intel, Live 0xf09db000
aes_i586 16995 1 aesni_intel, Live 0xf092d000
snd_seq_midi_event 14475 1 snd_seq_midi, Live 0xf0960000
xts 12778 1 aesni_intel, Live 0xf0942000
gf128mul 14503 2 lrw,xts, Live 0xf0948000
snd_seq 55716 2 snd_seq_midi,snd_seq_midi_event, Live 0xf097e000
snd_timer 28930 3 snd_pcm,snd_seq, Live 0xf094f000
snd_seq_device 14137 3 snd_seq_midi,snd_rawmidi,snd_seq, Live 0xf093d000
microcode 18938 0 - Live 0xf0879000
snd 61270 14
```

Miscs

gd@gdbox:/proc$ cat cmdline
BOOT_IMAGE=/boot/vmlinuz-3.11.0-15-generic root=UUID=ac2983b8-83f7-4919-af5f-ab7c09012432 ro quiet splash vt.handoff=7

gd@gdbox:/proc$ cat partitions

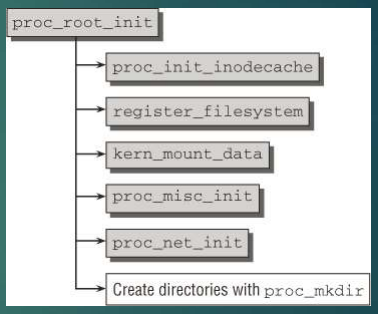
major	minor	#blocks	name
11	0	56868	sr0
8	0	33554432	sda
8	1	32768000	sda1
8	2	1	sda2
8	5	783360	sda5

81

初始化

```
fs/proc_root.c
struct proc_dir_entry *proc_net, *proc_bus,
*proc_root_fs, *proc_root_driver;

void __init proc_root_init(void)
{
...
        proc_net = proc_mkdir("sysvipc", NULL);
...
        proc_root_fs = proc_mkdir("fs", NULL);
        proc_root_driver = proc_mkdir("driver", NULL);
...
        proc_bus = proc_mkdir("bus", NULL);
}
```



82

创建新的proc文件

```
static struct proc_dir_entry *proc_lll_entry = NULL ;

static const struct file_operations proc_lll_fops = {
 .owner = THIS_MODULE,
 .read  = proc_lll_read,
};


 /* Create /proc/llaolao */
    proc_lll_entry = proc_create("llaolao", 0, NULL, &proc_lll_fops);


 if(proc_lll_entry)
        proc_remove(proc_lll_entry);
```



```
static ssize_t proc_lll_read(struct file *filp, char __user * buf, size_t count, loff_t * offp)
{
    int n = 0, ret;
    char secrets[100];

    printk(KERN_INFO "proc_lll_read is called file %p, buf %p count %d off %llx\n",
        filp, buf, count, *offp);
    sprintf(secrets, "kernel secrets balabala %s...\n", filp->f_path.dentry->d_iname);

    n = strlen(secrets);
    if(*offp < n)
    {
          copy_to_user(buf, secrets, n+1);
        *offp = n+1;
        ret = n+1;
    }
    else
        ret = 0;

    return ret;
}

[ 5721.084685] proc_lll_read is called file d9026cc0, buf 09eb4000 count 32768 off 0
[ 5721.085419] proc_lll_read is called file d9026cc0, buf 09eb4000 count 32768 off 24
```

Seq File

```
/*
 * linux/fs/seq_file.c
 *
 * helper functions for making synthetic files from sequences of records.
 * initial implementation -- AV, Oct 2001.
 */
```

- An iterator interface which lets a virtual file implementation step through the objects it is presenting
- Utility functions for formatting objects for output
- Canned file_operations which implement most operations on the virtual file

al viro, chuck, and niels
USENIX 2000

85

(This, btw, is something that Al Viro does absolutely beautifully. I don't know how many people look at Al's progression of patches, but they are stand-alone patches on their own, while at the same time _also_ being part of a larger migration to the inscrutable goals of Al - ie namespaces etc. You may not realize just _how_ impressive that is, and what a absolute wonder it is to work with the guy. Poetry in patches, indeed).

Date: Thu, 27 Dec 2001 12:21:02 -0800 (PST)
From: Linus Torvalds <torvalds@transmeta.com>
Subject: Re: The direction linux is taking

86

在Sun Apr 15 2018，Linus宣布4.17rc的邮件中，Al有7个更新，名字中的A帮他占据首位

...
Go out and test,

Linus

Al Viro (7):
vfs dcache updates
misc vfs updates
sparc syscall cleanups
alpha syscall cleanups
vfs namei updates
AFS updates
vfs thaw updates

http://lkml.iu.edu/hypermail/linux/kernel/1804.1/06654.html

87

```
static int version_proc_show(struct seq_file *m, void *v)
{
    seq_printf(m, linux_proc_banner,
        utsname()->sysname,
        utsname()->release,
        utsname()->version);
    return 0;
}

static int version_proc_open(struct inode *inode, struct file *file)
{
    return single_open(file, version_proc_show, NULL);
}

static const struct file_operations version_proc_fops = {
    .open        = version_proc_open,
    .read        = seq_read,
    .llseek      = seq_lseek,
    .release = single_release,
};

static int __init proc_version_init(void)
{
    proc_create("version", 0, NULL, &version_proc_fops);
    return 0;
}
fs_initcall(proc_version_init);
```

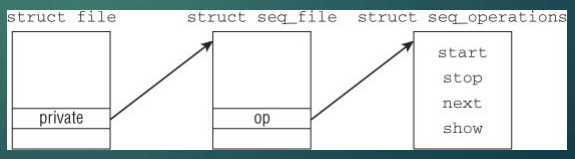
88

```
(gdb) bt
#0  seq_open (file=0xe34efb40, op=0xc169d5c4) at fs/seq_file.c:81
#1  0xc11b02a1 in mounts_open_common (inode=<optimized out>, file=0xe34efb40,
   show=0xc11affb0 <show_vfsmnt>) at fs/proc_namespace.c:267
#2  0xc11b03a2 in mounts_open (inode=<optimized out>, file=<optimized out>)
   at fs/proc_namespace.c:298
#3  0xc1175f7e in do_dentry_open (f=0xe34efb40, open=0xc11b0390 <mounts_open>,
   cred=0xeec0a300) at fs/open.c:708
#4  0xc11761f2 in finish_open (file=<optimized out>, dentry=<optimized out>,
   open=0, opened=0xe358dedc) at fs/open.c:774
#5  0xc1185633 in do_last (nd=0xe358df00, path=0xe358dec8, file=0xe34efb40,
   op=0xe358df80, opened=0xe358dedc, name=0xeec87000) at fs/namei.c:3068
#6  0xc1186055 in path_openat (dfd=<optimized out>, pathname=0xeec87000,
   nd=0xe358df00, op=0xe358df80, flags=65) at fs/namei.c:3228
#7  0xc1186645 in do_filp_open (dfd=-100, pathname=0xeec87000, op=0xe358df80)
   at fs/namei.c:3259
#8  0xc11775f5 in do_sys_open (dfd=-100, filename=<optimized out>,
   flags=524288, mode=438) at fs/open.c:970
#9  0xc1177702 in SYSC_open (mode=<optimized out>, flags=<optimized out>,
   filename=<optimized out>) at fs/open.c:988
#10 SyS_open (filename=-1222720991, flags=524288, mode=<optimized out>)
   at fs/open.c:983
#11 <signal handler called>
#12 0xb773c424 in ?? ()
```


/include/linux/seq_file.h

```
struct seq_file {
        char *buf;
        size_t size;
        size_t from;
        size_t count;
        size_t pad_until;
        loff_t index;
        loff_t read_pos;
        u64 version;
        struct mutex lock;
        const struct seq_operations *op;
        int poll_event;
#ifdef CONFIG_USER_NS
        struct user_namespace *user_ns;
#endif
        void *private;
};
```

```
struct seq_operations {
        void * (*start) (struct seq_file *m, loff_t *pos);
        void (*stop) (struct seq_file *m, void *v);
        void * (*next) (struct seq_file *m, void *v, loff_t *pos
        int (*show) (struct seq_file *m, void *v);
};
```



格物

```
(gdb) print p
$1 = (struct seq_file *) 0xd8864a80
(gdb) print *p
$2 = {buf = 0x0, size = 0, from = 0, count = 0, index = 0, read_pos = 0,
  version = 0, lock = {count = {counter = 1}, wait_lock = {{rlock = {
        raw_lock = {{head_tail = 0, tickets = {head = 0 '\000',
              tail = 0 '\000'}}}}}, wait_list = {next = 0xd8864ab0,
     prev = 0xd8864ab0}, owner = 0x0, spin_mlock = 0x0}, op = 0xc169d5c4,
  poll_event = 0, private = 0x0}
```

```
(gdb) print *p->op
$3 = {start = 0xc11922c0 <m_start>, stop = 0xc11922a0 <m_stop>,
  next = 0xc1192280 <m_next>, show = 0xc1192220 <m_show>}
```


m_start

```
(gdb) print priv
$4 = (struct proc_maps_private *) 0xeec6e150
(gdb) print *priv
$5 = {pid = 0xd1692d40, task = 0x0, tail_vma = 0x0}
```

```
(gdb) bt
#0  d_path (path=0xd7e2f188, buf=0xeec860af "", buflen=3921)
    at fs/dcache.c:3056
#1  0xc1196d55 in seq_path (m=0xd8949840, path=<optimized out>,
    esc=0xc18c13a6 "\n") at fs/seq_file.c:471
#2  0xc11c9f78 in show_map_vma (m=0xd8949840, vma=0xd7fd15a0, is_pid=1)
    at fs/proc/task_mmu.c:305
#3  0xc11ca13a in show_map (is_pid=1, v=0xd7fd15a0, m=0xd8949840)
    at fs/proc/task_mmu.c:356
#4  show_pid_map (m=<optimized out>, v=<optimized out>)
    at fs/proc/task_mmu.c:366
#5  0xc1196753 in seq_read (file=0xdd809d80,
    buf=0x9a56000 <Address 0x9a56000 out of bounds>, size=32768,
    ppos=0xd892bf98) at fs/seq_file.c:256
#6  0xc11787fc in vfs_read (file=0xdd809d80,
    buf=0x9a56000 <Address 0x9a56000 out of bounds>, count=32768,
    pos=0xd892bf98) at fs/read_write.c:396
#7  0xc1178a57 in SYSC_read (count=32768,
    buf=0x9a56000 <Address 0x9a56000 out of bounds>, fd=<optimized out>)
    at fs/read_write.c:506
#8  SyS_read (fd=3, buf=161832960, count=32768) at fs/read_write.c:499
```

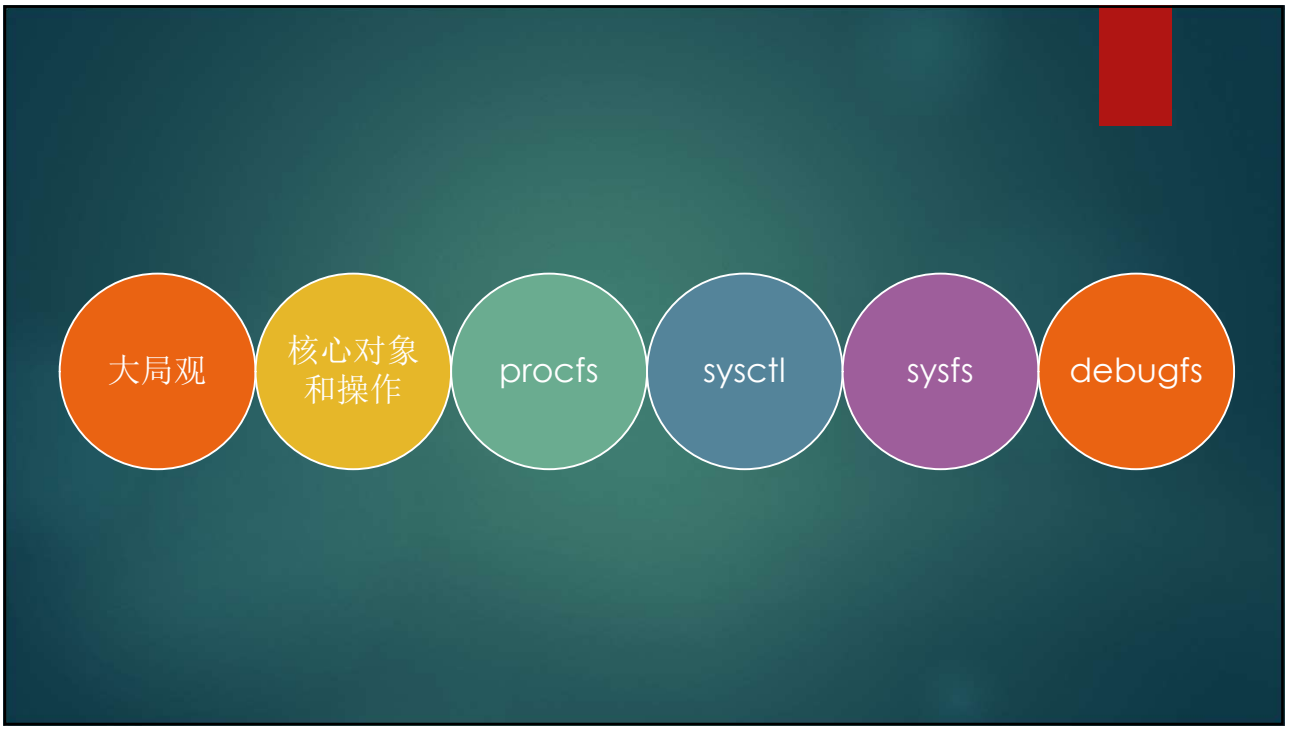


```
(gdb) p vma->vm_file->f_path->dentry->d_iname
$42 = "syslogd\00


(gdb) p vma->vm_file->f_path->dentry->d_iname
$47 = "libnss_files-2.15.so\00



(gdb) p /x vma->vm_start
$51 = 0xb759f000
(gdb) p /x vma->vm_end
$52 = 0xb75aa000
```



95

Basics

- A means of configuring certain aspects of the kernel at run-time
- The directory consists of several (arch-dependent) subdirs.
- Each subdir is mainly about one part of the kernel,

```
ge@gewubox:/proc/sys$ tree -L 2
.
├── debug
│   ├── exception-trace
│   └── kprobes-optimization
├── dev
│   ├── cdrom
│   ├── hpet
│   ├── mac_hid
│   ├── parport
│   ├── raid
│   └── scsi
├── fs
│   ├── aio-max-nr
│   ├── aio-nr
│   ├── binfmt_misc
│   ├── dentry-state
│   ├── dir-notify-enable
│   ├── epoll
│   ├── file-max
│   ├── file-nr
│   ├── inode-nr
│   ├── inode-state
│   ├── inotify
│   ├── lease-break-time
│   ├── leases-enable
│   ├── mqueue
│   ├── nr_open
│   ├── overflowgid
│   ├── overflowuid
│   ├── pipe-max-size
│   ├── protected_hardlinks
│   ├── protected_symlinks
│   ├── quota
│   └── suid_dumpable
├── kernel
│   ├── acct
│   ├── acpi_video_flags
│   ├── auto_msgmni
```

96

文档

Lister - [D:\bench\linux-3.16.3\Documentation\sysctl\README]

File Edit Options Help

```
Documentation for /proc/sys/            kernel version 2.2.10
        (c) 1998, 1999,  Rik van Riel <riel@nl.linux.org>

'Why', I hear you ask, 'would anyone even _want_ documentation
for them sysctl files? If anybody really needs it, it's all in
the source...'

Well, this documentation is written because some people either
don't know they need to tweak something, or because they don't
have the time or knowledge to read the source code.

Furthermore, the programmers who built sysctl have built it to
be actually used, not just for the fun of programming it :-)

==============================================================
```


睿智

```
Legal blurb:

As usual, there are two main things to consider:
1. you get what you pay for
2. it's free

The consequences are that I won't guarantee the correctness of
this document, and if you come to me complaining about how you
screwed up your system because of wrong documentation, I won't
feel sorry for you. I might even laugh at you...

But of course, if you _do_ manage to screw up your system using
only the sysctl options used in this file, I'd like to hear of
it. Not only to have a great laugh, but also to make sure that
you're the last RTFMing person to screw up.

In short, e-mail your suggestions, corrections and / or horror
stories to: <riel@nl.linux.org>

Rik van Riel.

==============================================================
```

文档

d:\bench\linux-3.16.3\Documentation\sysctl*.*

↑Name	Ext	Size	Date	Attr
[..]		<DIR>	11/25/2014 10:07	——
00-index		384	09/18/2014 01:22	-a—
abi	txt	1,467	09/18/2014 01:22	-a—
fs	txt	11,898	09/18/2014 01:22	-a—
kernel	txt	31,728	09/18/2014 01:22	-a—
net	txt	9,523	09/18/2014 01:22	-a—
Readme		2,750	09/18/2014 01:22	-a—
sunrpc	txt	804	09/18/2014 01:22	-a—
vm	txt	30,785	09/18/2014 01:22	-a—

示例

```
root@gewubox:/proc/sys# cat /proc/sys/kernel/printk
4       4       1       7
```

```
root@gewubox:/proc/sys/kernel# ls
acct                    msgmni                  sched_latency_ns
acpi_video_flags        msg_next_id             sched_migration_cost_ns
auto_msgmni             ngroups_max             sched_min_granularity_ns
blk_iopoll              nmi_watchdog            sched_nr_migrate
bootloader_type         ns_last_pid             sched_rr_timeslice_ms
bootloader_version      osrelease               sched_rt_period_us
cad_pid                 ostype                  sched_rt_runtime_us
cap_last_cap            overflowgid             sched_shares_window_ns
core_pattern            overflowuid             sched_time_avg_ms
core_pipe_limit         panic                   sched_tunable_scaling
core_uses_pid           panic_on_io_nmi         sched_wakeup_granularity_ns
ctrl-alt-del            panic_on_oops           sem
dmesg_restrict          panic_on_unrecovered_nmi sem_next_id
domainname              perf_cpu_time_max_percent sg-big-buff
ftrace_dump_on_oops     perf_event_max_sample_rate shmall
ftrace_enabled          perf_event_mlock_kb     shmmax
hostname                perf_event_paranoid     shmmni
hotplug                 pid_max                 shm_next_id
hung_task_check_count   poweroff_cmd            shm_rmid_forced
hung_task_panic         print-fatal-signals     softlockup_panic
hung_task_timeout_secs  printk                  stack_tracer_enabled
hung_task_warnings      printk_delay            sysrq
io_delay_type           printk_ratelimit        tainted
keys                    printk_ratelimit_burst  threads-max
kptr_restrict           pty                     timer_migration
kstack_depth_to_print   random                  traceoff_on_warning
latencytop              randomize_va_space      unknown_nmi_panic
max_lock_depth          real-root-dev           usermodehelper
modprobe                sched_autogroup_enabled version
modules_disabled        sched_cfs_bandwidth_slice_us watchdog
msgmax                  sched_child_runs_first  watchdog_thresh
msgmnb                  sched_domain            yama
```

sysctl命令

ge@gewubox:~$ sysctl
usage: sysctl [-n] [-e] variable ...
    sysctl [-n] [-e] [-q] -w variable=value ...
    sysctl [-n] [-e] -a
    sysctl [-n] [-e] [-q] -p <file> (default /etc/sysctl.conf)
    sysctl [-n] [-e] -A

```
ge@gewubox:~$ sudo sysctl -w kernel.hostname=gedubox
[sudo] password for ge:
kernel.hostname = gedubox
ge@gewubox:~$ sysctl -n kernel.hostname
gedubox
```

gedu@gedu-VirtualBox:~$ sudo sysctl -w kernel.sysrq=1
kernel.sysrq = 1

实现

```
int __init proc_sys_init(void)
{
    struct proc_dir_entry *proc_sys_root;

    proc_sys_root = proc_mkdir("sys", NULL);
    proc_sys_root->proc_iops = &proc_sys_dir_operations;
    proc_sys_root->proc_fops = &proc_sys_dir_file_operations;
    proc_sys_root->nlink = 0;

    return sysctl_init();
}
```

服务接口

```
/**
 * register_sysctl - register a sysctl table
 * @path: The path to the directory the sysctl table is in.
 * @table: the table structure
 *
 * Register a sysctl table. @table should be a filled in ctl_table
 * array. A completely 0 filled entry terminates the table.
 *
 * See __register_sysctl_table for more details.
 */
struct ctl_table_header *register_sysctl(const char *path, struct ctl_table *table)
{
        return __register_sysctl_table(&sysctl_table_root.default_set,
                                       path, table);
}
EXPORT_SYMBOL(register_sysctl);
```


使用者/kernel/sysctl.c

```
int __init sysctl_init(void)
{
        struct ctl_table_header *hdr;

        hdr = register_sysctl_table(sysctl_base_table);
        kmemleak_not_leak(hdr);
        return 0;
}
```

```
static struct ctl_table kern_table[] = {
        {
                .procname   = "sched_child_runs_first",
                .data       = &sysctl_sched_child_runs_first,
                .maxlen     = sizeof(unsigned int),
                .mode       = 0644,
                .proc_handler   = proc_dointvec,
        },
#ifdef CONFIG_SCHED_DEBUG
        {
                .procname   = "sched_min_granularity_ns",
```

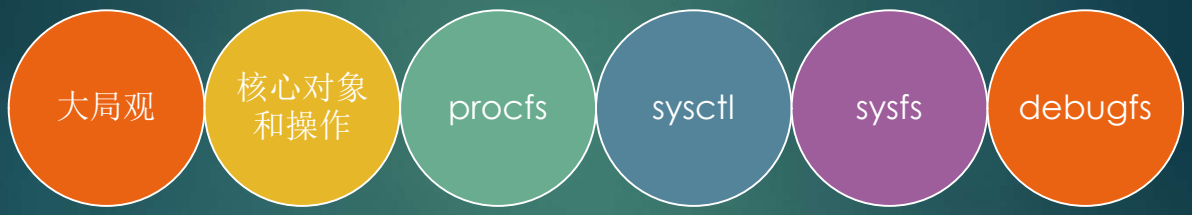
```
/* The default sysctl tables: */

static struct ctl_table sysctl_base_table[] = {
        {
                .procname       = "kernel",
                .mode           = 0555,
                .child          = kern_table,
        },
        {
                .procname       = "vm",
                .mode           = 0555,
                .child          = vm_table,
        },
        {
                .procname       = "fs",
                .mode           = 0555,
                .child          = fs_table,
        },
        {
                .procname       = "debug",
                .mode           = 0555,
                .child          = debug_table,
        },
        {
                .procname       = "dev",
                .mode           = 0555,
                .child          = dev_table,
        },
        { }
};
```

sysctl与普通procfs比较

- Both procfs and sysctl export kernel-internal information, but procfs mainly exports read-only data, while most sysctl information is writable too (but only by the superuser).
- As far as exporting read-only data, the choice between procfs and sysctl depends on how much information is supposed to be exported.
- Files associated with a simple kernel variable or data structure are exported with sysctl. The others, which are associated with more complex data structures and may need special formatting, are exported with procfs. Examples of the latter category are caches and statistics.

105



106

What it's

- Sysfs is an in-memory filesystem to export kernel data structures, their attributes, and the linkages between them to userspace.
- sysfs is tied inherently to the kobject infrastructure
- It's mounted under /sys at boot time (look at /etc/fstab for the specifier)

```
sysfs - _The_ filesystem for exporting kernel objects.

Patrick Mochel  <mochel@osdl.org>
Mike Murphy <mamurph@cs.clemson.edu>

Revised:    16 August 2011
Original:   10 January 2003
```

mount -t sysfs sysfs /sys

Directory Creation

- For every kobject that is registered with the system, a directory is created for it in sysfs.
- That directory is created as a subdirectory of the kobject's parent, expressing internal object hierarchies to userspace.
- Top-level directories in sysfs represent the common ancestors of object hierarchies; i.e. the subsystems the objects belong to.

File Creation

```
int sysfs_create_file(struct kobject * kobj, const struct attribute * attr);
void sysfs_remove_file(struct kobject * kobj, const struct attribute * attr);

struct attribute {
        char                    * name;
        struct module           *owner;
        umode_t                 mode;
};
```


Read/Write Attribute Data

```
struct sysfs_ops {
        ssize_t (*show)(struct kobject *, struct attribute *, char *);
        ssize_t (*store)(struct kobject *, struct attribute *, const char *, size_t);
};
```

- Subsystems should define a struct kobj_type as a descriptor for this type, which is where the sysfs_ops pointer is stored.

包装/派生 —— 设备属性

```
struct device_attribute {
    struct attribute  attr;
    ssize_t (*show)(struct device *dev, struct device_attribute *attr,
            char *buf);
    ssize_t (*store)(struct device *dev, struct device_attribute *attr,
            const char *buf, size_t count);
};

int device_create_file(struct device *, const struct device_attribute *);
void device_remove_file(struct device *, const struct device_attribute *);
```


续

```
#define DEVICE_ATTR(_name, _mode, _show, _store) \
struct device_attribute dev_attr_##_name = __ATTR(_name, _mode, _show, _store)
```

```
ssize_t (*show)(struct device *dev, struct device_attribute *attr, char *buf);
ssize_t (*store)(struct device *dev, struct device_attribute *attr,
            const char *buf, size_t count);
```

- To read or write attributes, show() or store() methods must be specified when declaring the attribute.

示例

```
static ssize_t show_name(struct device *dev, struct device_attribute *attr,
                         char *buf)
{
    return scnprintf(buf, PAGE_SIZE, "%s\n", dev->name);
}

static ssize_t store_name(struct device *dev, struct device_attribute *attr,
                          const char *buf, size_t count)
{
    snprintf(dev->name, sizeof(dev->name), "%.*s",
             (int)min(count, sizeof(dev->name) - 1), buf);
    return count;
}

static DEVICE_ATTR(name, S_IRUGO, show_name, store_name);
```


包装/派生 —— 总线属性

```
struct bus_attribute {
        struct attribute        attr;
        ssize_t (*show)(struct bus_type *, char * buf);
        ssize_t (*store)(struct bus_type *, const char * buf, size_t count);
};
```

Declaring:
BUS_ATTR(_name, _mode, _show, _store)

Creation/Removal:
int bus_create_file(struct bus_type *, struct bus_attribute *);
void bus_remove_file(struct bus_type *, struct bus_attribute *);

包装/派生 —— 驱动属性

```
struct driver_attribute {
        struct attribute        attr;
        ssize_t (*show)(struct device_driver *, char * buf);
        ssize_t (*store)(struct device_driver *, const char * buf,
                        size_t count);
};
```

Declaring:
DRIVER_ATTR(_name, _mode, _show, _store)

Creation/Removal:
int driver_create_file(struct device_driver *, const struct driver_attribute *);
void driver_remove_file(struct device_driver *, const struct driver_attribute *);

Top Level Directory

block/
bus/
class/
dev/
devices/
firmware/
net/
fs/

```
gd@gdbox:/sys$ ls
block  class  devices   fs          kernel  power
bus    dev    firmware  hypervisor  module
```

写给应用程序开发者的话

Lister - [D:\bench\linux-3.16.3\Documentation\sysfs-rules.txt]

File Edit Options Help

```
Rules on how to access information in the Linux kernel sysfs

The kernel-exported sysfs exports internal kernel implementation details
and depends on internal kernel structures and layout. It is agreed upon
by the kernel developers that the Linux kernel does not provide a stable
internal API. Therefore, there are aspects of the sysfs interface that
may not be stable across kernel releases.

To minimize the risk of breaking users of sysfs, which are in most cases
low-level userspace applications, with a new kernel release, the users
of sysfs must follow some rules to use an as-abstract-as-possible way to
access this filesystem. The current udev and HAL programs already
implement this and users are encouraged to plug, if possible, into the
abstractions these programs provide instead of accessing sysfs directly.

But if you really do want or need to access sysfs directly, please follow
the following rules and then your programs should work with future
versions of the sysfs interface.

- Do not use libsysfs
  It makes assumptions about sysfs which are not true. Its API does not
```


大局观

核心对象和操作

procfs

sysctl

sysfs

debugfs

What it's

Lister - [D:\bench\linux-3.16.3\Documentation\filesystems\debugfs.txt]

File Edit Options Help

```
Copyright 2009 Jonathan Corbet <corbet@lwn.net>

Debugfs exists as a simple way for kernel developers to make information
available to user space.  Unlike /proc, which is only meant for information
about a process, or sysfs, which has strict one-value-per-file rules,
debugfs has no rules at all.  Developers can put any information they want
there.  The debugfs filesystem is also intended to not serve as a stable
ABI to user space; in theory, there are no stability constraints placed on
files exported there.  The real world is not always so simple, though [1];
even debugfs interfaces are best designed with the idea that they will need
to be maintained forever.

Debugfs is typically mounted with a command like:

    mount -t debugfs none /sys/kernel/debug
```

"The debugfs filesystem allows for implementing a bidirectional debugging interface with only a few function calls." -- *Professional Linux Kernel Architecture* (10.2 Simple Filesystems))

libfs

- a library that provides several very generic standard routines that can be used to create small filesystems
- The routines are well suited for in-memory files without a backing store
- The library code is contained in a single file, fs/libfs.c
  - The prototypes are defined in <fs.h>; there is *no* <libfs.h>!
- Routines provided by libfs are generally prefixed by simple_
- Debugfs is one filesystem that employs libfs

<linux/debugfs.h>

```
struct dentry *debugfs_create_file(const char *name, mode_t mode,
                                   struct dentry *parent, void *data,
                                   const struct file_operations *fops);

struct dentry *debugfs_create_dir(const char *name, struct dentry *parent);

struct dentry *debugfs_create_symlink(const char *name, struct dentry *parent,
                                      const char *dest);


void debugfs_remove(struct dentry *dentry);

struct dentry *debugfs_rename(struct dentry *old_dir, struct dentry *old_dentry,
                              struct dentry *new_dir, const char *new_name);

struct dentry *debugfs_create_XX(const char *name, mode_t mode,
                                 struct dentry *parent, XX *value);
```



```
ge@gewubox:~/work/linux-3.12.2/drivers$ grep "debugfs_create_dir" . -R
./mmc/host/s3cmci.c:    host->debug_root = debugfs_create_dir(dev_name(dev), NULL);
./mmc/core/debugfs.c:   root = debugfs_create_dir(mmc_hostname(host), NULL);
./mmc/core/debugfs.c:   root = debugfs_create_dir(mmc_card_id(card), host->debugfs_root);
./remoteproc/remoteproc_debugfs.c:      rproc->dbg_dir = debugfs_create_dir(dev_name(dev), rproc_dbg);
./remoteproc/remoteproc_debugfs.c:              rproc_dbg = debugfs_create_dir(KBUILD_MODNAME, NULL);
./input/touchscreen/edt-ft5x06.c:       tsdata->debug_dir = debugfs_create_dir(debugfs_name, NULL);
./block/pktcdvd.c:      pd->dfs_d_root = debugfs_create_dir(pd->name, pkt_debugfs_root);
./block/pktcdvd.c:      pkt_debugfs_root = debugfs_create_dir(DRIVER_NAME, NULL);
./block/rsxx/core.c:    card->debugfs_dir = debugfs_create_dir(card->gendisk->disk_name, NULL);
./block/aoe/aoeblk.c:   aoe_debugfs_dir = debugfs_create_dir("aoe", NULL);
./block/mtip32xx/mtip32xx.c:    dd->dfs_node = debugfs_create_dir(dd->disk->disk_name, dfs_parent);
./block/mtip32xx/mtip32xx.c:    dfs_parent = debugfs_create_dir("rssd", NULL);
./char/virtio_console.c:        pdrvdata.debugfs_dir = debugfs_create_dir("virtio-ports", NULL);
./net/wireless/cw1200/debug.c:  d->debugfs_phy = debugfs_create_dir("cw1200",
./net/wireless/rt2x00/rt2x00debug.c:        debugfs_create_dir(intf->rt2x00dev->ops->name,
./net/wireless/rt2x00/rt2x00debug.c:        debugfs_create_dir("register", intf->driver_folder);
./net/wireless/rt2x00/rt2x00debug.c:        debugfs_create_dir("queue", intf->driver_folder);
./net/wireless/mwifiex/debugfs.c:       priv->dfs_dev_dir = debugfs_create_dir(priv->netdev->name,
./net/wireless/mwifiex/debugfs.c:               mwifiex_dfs_dir = debugfs_create_dir("mwifiex", NULL);
./net/wireless/iwlwifi/iwl-drv.c:       drv->dbgfs_op_mode = debugfs_create_dir(op->name,
./net/wireless/iwlwifi/iwl-drv.c:       drv->dbgfs_drv = debugfs_create_dir(dev_name(trans->dev),
./net/wireless/iwlwifi/iwl-drv.c:       drv->trans->dbgfs_dir = debugfs_create_dir("trans", drv->dbgfs_drv);
./net/wireless/iwlwifi/iwl-drv.c:       iwl_dbgfs_root = debugfs_create_dir(DRV_NAME, NULL);
./net/wireless/iwlwifi/mvm/debugfs.c:   mvmvif->dbgfs_dir = debugfs_create_dir("iwlmvm", dbgfs_dir);
./net/wireless/iwlwifi/dvm/debugfs.c:   dir_data = debugfs_create_dir("data", dbgfs_dir);
./net/wireless/iwlwifi/dvm/debugfs.c:   dir_rf = debugfs_create_dir("rf", dbgfs_dir);
./net/wireless/iwlwifi/dvm/debugfs.c:   dir_debug = debugfs_create_dir("debug", dbgfs_dir);
./net/wireless/libertas/debugfs.c:              lbs_dir = debugfs_create_dir("lbs_wireless", NULL);
./net/wireless/libertas/debugfs.c:      priv->debugfs_dir = debugfs_create_dir(dev->name, lbs_dir);
./net/wireless/libertas/debugfs.c:      priv->events_dir = debugfs_create_dir("subscribed_events", priv->debugfs_dir);
./net/wireless/libertas/debugfs.c:      priv->regs_dir = debugfs_create_dir("registers", priv->debugfs_dir);
./net/wireless/brcm80211/brcmfmac/dhd_dbg.c:    root_folder = debugfs_create_dir(KBUILD_MODNAME, NULL);
```

/drivers/idle/i7300_idle.c

```
static ssize_t stats_read_ul(struct file *fp, char __user *ubuf, size_t count,
                             loff_t *off)
{
        unsigned long *p = fp->private_data;
        char buf[32];
        int len;

        len = snprintf(buf, 32, "%lu\n", *p);
        return simple_read_from_buffer(ubuf, count, off, buf, len);
}

static const struct file_operations idle_fops = {
        .open       = simple_open,
        .read       = stats_read_ul,
        .llseek = default_llseek,
};

struct debugfs_file_info {
        void *ptr;
        char name[32];
        struct dentry *file;
} debugfs_file_list[] = {
                        {&total_starts, "total_starts", NULL},
                        {&total_us, "total_us", NULL},
#ifdef DEBUG
                        {&past_skip, "past_skip", NULL},
#endif
                        {NULL, "", NULL}
                    };
```

```
debugfs_dir = debugfs_create_dir("i7300_idle", NULL);
if (debugfs_dir) {
        int i = 0;

        while (debugfs_file_list[i].ptr != NULL) {
                debugfs_file_list[i].file = debugfs_create_file(
                        debugfs_file_list[i].name,
                        S_IRUSR,
                        debugfs_dir,
                        debugfs_file_list[i].ptr,
                        &idle_fops);
                i++;
        }
}
```


simple attribute files

/include/linux/fs.h

```
/*
 * simple attribute files
 *
 * These attributes behave similar to those in sysfs:
 *
 * Writing to an attribute immediately sets a value, an open file can be
 * written to multiple times.
 *
 * Reading from an attribute creates a buffer from the value that might get
 * read with multiple read calls. When the attribute has been read
 * completely, no further read calls are possible until the file is opened
 * again.
 *
 * All attributes contain a text representation of a numeric value
 * that are accessed with the get() and set() functions.
 */
```

DEFINE_SIMPLE_ATTRIBUTE

/include/linux/fs.h

```
#define DEFINE_SIMPLE_ATTRIBUTE(__fops, __get, __set, __fmt)        \
static int __fops ## _open(struct inode *inode, struct file *file)   \
{                                                                    \
        __simple_attr_check_format(__fmt, 0ull);                     \
        return simple_attr_open(inode, file, __get, __set, __fmt);   \
}                                                                    \
static const struct file_operations __fops = {                       \
        .owner   = THIS_MODULE,                                      \
        .open    = __fops ## _open,                                  \
        .release = simple_attr_release,                              \
        .read    = simple_attr_read,                                 \
        .write   = simple_attr_write,                                \
        .llseek  = generic_file_llseek,                              \
};
```



```
static int
i915_max_freq_get(void *data, u64 *val)
{
        struct drm_device *dev = data;
        struct drm_i915_private *dev_priv = dev->dev_private;
        int ret;

        ret = mutex_lock_interruptible(&dev_priv->rps.hw_lock);
        *val = vlv_gpu_freq(dev_priv, dev_priv->rps.max_freq_softlimit);
        mutex_unlock(&dev_priv->rps.hw_lock);

        return 0;
}

static int
i915_max_freq_set(void *data, u64 val)
{
        struct drm_device *dev = data;
        struct drm_i915_private *dev_priv = dev->dev_private;
        u32 rp_state_cap, hw_max, hw_min;
        int ret;
        //...
        dev_priv->rps.max_freq_softlimit = val;
        //...
```

```
DEFINE_SIMPLE_ATTRIBUTE(i915_max_freq_fops,
                        i915_max_freq_get,
i915_max_freq_set,   "%llu\n");
```

/drivers/gpu/drm/i915/i915_debugfs.c

```
// static varibales for debugfs
static struct dentry *df_dir = NULL, * df_dir;

static int
lll_age_set(void *data, u64 val)
{
    struct lll_profile_struct * lp = (struct lll_profile_struct *)data;
    lp->age = val;

    return 0;
}
static int
lll_age_get(void *data, u64 *val)
{
    struct lll_profile_struct * lp = (struct lll_profile_struct *)data;
    *val = lp->age;

    return 0;
}

// macro from linux/fs.h
DEFINE_SIMPLE_ATTRIBUTE(df_age_fops, lll_age_get, lll_age_set, "%llu\n");
```

127

```
df_dir = debugfs_create_dir("llaolao", 0);
if(!df_dir)
{
    printk(KERN_ERR "create dir under debugfs failed\n");
    return -1;
}

debugfs_create_file("age", 0222, df_dir, &lll_profile, &df_age_fops);


if(df_dir)
    // clean up all debugfs entries
    debugfs_remove_recursive(df_dir);
```

```
root@gewubox:/home/ge/work/llaolao2# sudo su
root@gewubox:/home/ge/work/llaolao2# echo "9999" > /sys/kernel/debug/llaolao/age
root@gewubox:/home/ge/work/llaolao2# cat /sys/kernel/debug/llaolao/age
9999
```

128

Resources

- Creating Linux virtual filesystems. 2002
- <http://lwn.net/Articles/13325/>

- The Linux Virtual File-system Layer by Neil Brown. 1999
- <http://www.cse.unsw.edu.au/~neilb/oss/linux-commentary/vfs.html>

- A tour of the Linux VFS by Michael K. Johnson. 1996
- <http://www.tldp.org/LDP/khg/HyperNews/get/fs/vfstour.html>

- A small trail through the Linux kernel by Andries Brouwer. 2001
- <http://www.win.tue.nl/~aeb/linux/vfs/trail.html>

Q & A

张银奎

devfs

- In Linux 2.6, devfs is obsolete by udev, although minimal support is still available. For more information on udev, go to http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ.


```
#0  bio_init (bio=0xd135b780) at fs/bio.c:273
#1  0xc11a78eb in bio_alloc_bioset (gfp_mask=16, nr_iovecs=1, bs=0x0)
    at fs/bio.c:443
#2  0xc11a8128 in bio_kmalloc (nr_iovecs=1, gfp_mask=16)
    at include/linux/bio.h:246
#3  __bio_map_kern (gfp_mask=16, len=36, data=0xd1388400, q=0xd12f75f8)
    at fs/bio.c:1442
#4  bio_map_kern (q=0xd12f75f8, data=<optimized out>, len=36, gfp_mask=16)
    at fs/bio.c:1484
#5  0xc12e7e6a in blk_rq_map_kern (q=0xd12f75f8, rq=0xd13643c0,
    kbuf=<optimized out>, len=36, gfp_mask=16) at block/blk-map.c:309
#6  0xc1458c0b in scsi_execute (sdev=0xd1373800, cmd=0xd12dbe4c "\022",
    data_direction=<optimized out>, buffer=0xd1388400, bufflen=36,
    sense=0xd135b9c0 "", timeout=5125, retries=3, flags=0, resid=0xd12dbe34)
    at drivers/scsi/scsi_lib.c:220
#7  0xc1459bf0 in scsi_execute_req_flags (sdev=0xd1373800,
    cmd=0xd12dbe4c "\022", data_direction=2, buffer=0xd1388400, bufflen=36,
    sshdr=0xd12dbe2c, timeout=5125, retries=3, resid=0xd12dbe34, flags=0)
    at drivers/scsi/scsi_lib.c:270
#8  0xc145b660 in scsi_execute_req (bufflen=36, buffer=0xd1388400,
    cmd=0xd12dbe4c "\022", resid=0xd12dbe34, retries=3,
    timeout=<optimized out>, sshdr=0xd12dbe2c, data_direction=2,
    sdev=0xd1373800) at include/scsi/scsi_device.h:418
#9  scsi_probe_lun (bflags=<synthetic pointer>, inq_result=0xd1388400 "",
    sdev=0xd1373800, result_len=<optimized out>)
    at drivers/scsi/scsi_scan.c:574
#10 scsi_probe_and_add_lun (starget=0xd135fc00, lun=0, bflagsp=0x0,
    sdevp=0xd12dbe80, rescan=1, hostdata=0x0) at drivers/scsi/scsi_scan.c:1049
#11 0xc145ce7a in __scsi_add_device (shost=0xd1317800,
    channel=<optimized out>, id=<optimized out>, lun=0, hostdata=0x0)
    at drivers/scsi/scsi_scan.c:1530
#12 0xc148a43f in ata_scsi_scan_host (ap=0xd1338000, sync=1)
```