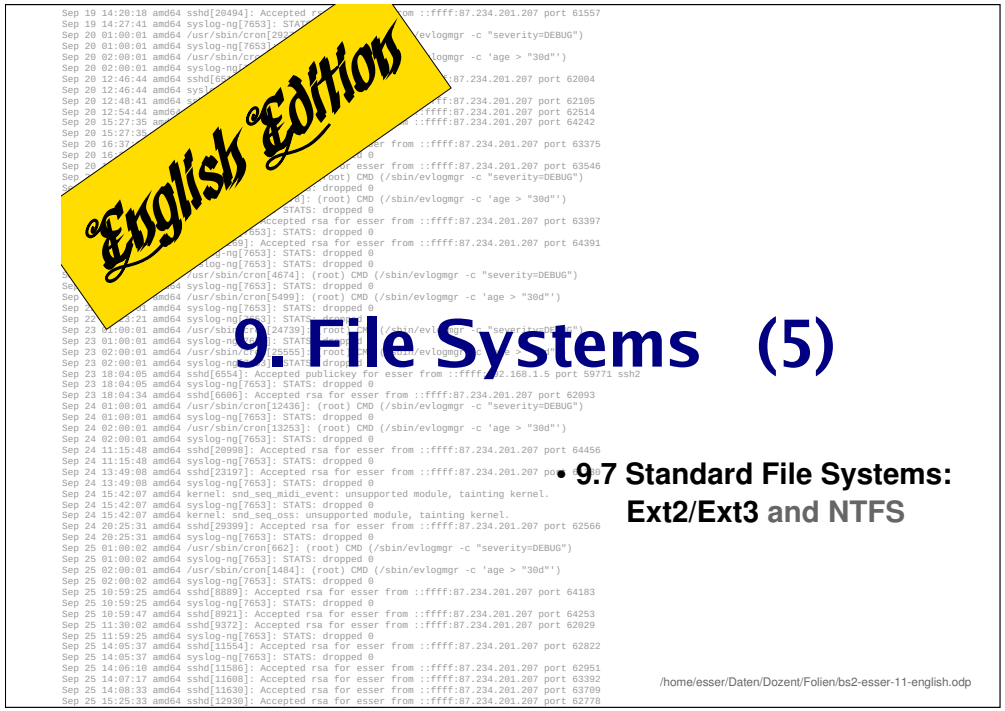



# 9. File Systems (5)

- **9.7 Standard File Systems: Ext2/Ext3 and NTFS**



# 9.7.1 Standard File Systems: Linux Ext2 / Ext3

## Ext2/Ext3 File System

- Ext3 is today's standard Linux file system (besides ReiserFS)
- History
  - in the beginning there was Minix...
  - Extended Filesystem replaced Minix
  - Ext2 (2nd Extended Filesystem) succeeded Ext
  - Ext3 (3rd Extended Filesystem) extends Ext2 with Journaling



## Ext2/Ext3: Features (1)

- Features
  - variable block sizes (depending on fs creation options, it handles many small or few big files well)
  - fast symbolic links (save link target directly in the inode when the address is short)
  - special attributes (which go beyond the traditional Unix file attributes), e.g. *immutable* flag
  - extended attributes, e.g. ACLs
  - compatibility bitmap for filesystem extensions (*read/write*-, *read-only*- und *incompatible*-Bits)

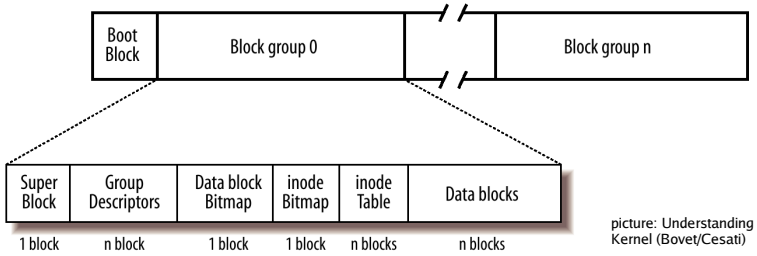
# Ext2/Ext3: Features (2)

- Features
  - guarantees regular filesystem checks, based on two properties:
    - mount count: mount filesystem only n times without check, then it must be fsck-ed
    - time since last fsck (maximal interval)
  - secure deletion (by overwriting the data blocks)
  - only Ext3: journaling
    - makes long fsck runs unnecessary in normal operation
    - speeds up fsck check in case of errors



# Ext2/Ext3: Design (1)

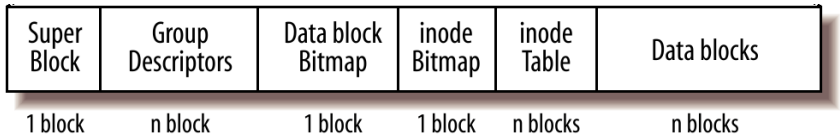
- design: partition is split into boot sector and **block groups**
- each block group holds a copy of the **superblock** (with administrative metadata for the whole filesystem)




picture: Understanding the Linux Kernel (Bovet/Cesati)



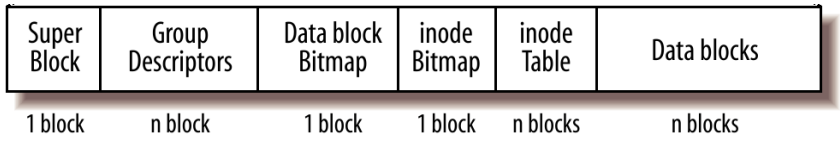
# Ext2/Ext3: Design (2)




- **group descriptor:** block group state (among other data: numbers of free blocks and inodes) (variable length)
  - each block group contains descriptors for *all* block groups
- **data block bitmap:** for each data block one bit (free/non-free)



# Ext2/Ext3: Design (3)




- **inode bitmap:** for each inode one bit: used/unused
- **inode table:** this contains all inodes of the block group (variable length)
- **data blocks:** the real data, files and directories

# Ext2/Ext3: Design (4)

- meta information in each block group – idea:
  - if the superblock is destroyed by a failure, there are redundant copies
  - small distance between meta informatio and data: shorter seek times when accessing the data
- real implementation works differently:
  - kernel keeps a superblock copy in RAM and only distributes it to the other groups when calling *fsck*
  - later Ext2 versions: *Sparse Superblock* option. superblocks only in groups 0, 1, 3^i, 5^i, 7^i (i>1)



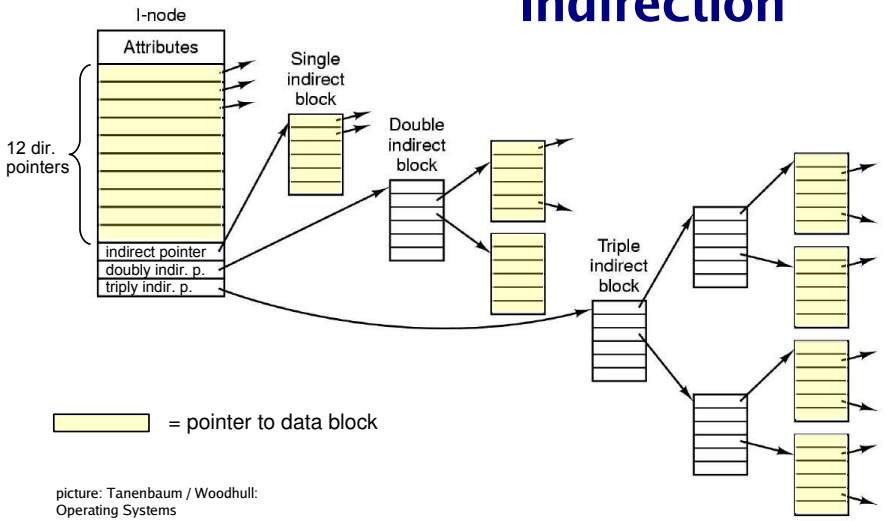
# File Sizes (1)

flexible in two ways:

- when creating the filesystem, several blocksizes are possible:
  - 1024 byte blocks -> max. size 16 GByte (x 2 to 2048 byte blocks, x 16)
  - 2048 byte blocks -> max. size 256 GByte (x 2 to 4096 byte blocks, x 16)
  - 4096 byte blocks -> max. size 4096 GByte (4 TB)
- for large files: save block numbers in indirection blocks (up to 3 layers):
  - doubling the block size = 16 times ($2^4$) as many block numbers: factor $2^3$ from indirection, factor 2 from block size



# File Sizes (2): Multiple Indirection




picture: Tanenbaum / Woodhull: Operating Systems



# File Sizes (3)

Example

length of a (disk) address: 4 Byte

| block size BS: | **1024 Byte** | **2048 Byte** | **4096 Byte** |
|---|---|---|---|
| 1 block can contain BS/4=N addresses | 256 | 512 | 1024 |
| 12 dir. point.: 12*BS = | 12 KByte | 24 KByte | 48 KByte |
| 1x indir. p.: 1 indir. bl. | $256^1$ x 1024 = 256 KB | $512^1$ x 2048 = 1 MB | $1024^1$ x 4096 = 4 MB |
| 2x indir. p.: N indir. bl. | $256^2$ x 1024 = 64 MB | $512^2$ x 2048 = 512 MB | $1024^2$ x 4096 = 4 GB |
| 3x indir. p.: $N^2$ indir. bl. | $256^3$ x 1024 = **16 GB** | $512^3$ x 2048 = **256 GB** | $1024^3$ x 4096 = **4 TB** |

numbers are theoretical; the maximum file size is restricted by other factors as well

# Ext2/Ext3 Superblock (1)

- drivers differentiate
  - structure on the **disk** (`ext2_super_block`)
  - structure in **main memory** (`ext2_sb_info`)
- on disk system uses fixed length data structures, e.g.
  - `__u32`: unsigned 32 bit
  - `__s16`: signed 16 bit
- in main memory system uses default datatypes (`unsigned long`, `int` etc.)



# Ext2/Ext3 Superblock (2)

- data structures on disk and in memory

| Type | Disk data structure | Memory data structure | Caching mode |
|---|---|---|---|
| Superblock | ext2_super_block | ext2_sb_info | Always cached |
| Group descriptor | ext2_group_desc | ext2_group_desc | Always cached |
| Block bitmap | Bit array in block | Bit array in buffer | Dynamic |
| inode bitmap | Bit array in block | Bit array in buffer | Dynamic |
| inode | ext2_inode | ext2_inode_info | Dynamic |
| Data block | Array of bytes | VFS buffer | Dynamic |
| Free inode | ext2_inode | None | Never |
| Free block | Array of bytes | None | Never |

table: Understanding the Linux Kernel (Bovet/Cesati)



# Ext2/Ext3 Superblock (3)

file: *include/linux/ext2_fs.h*

```
struct ext2_super_block {
    __le32  s_inodes_count;      /* Inodes count */
    __le32  s_blocks_count;      /* Blocks count */
```

how many blocks and inodes does the filesystem have?

```
    __le32  s_r_blocks_count;    /* Reserved blocks count */
    __le32  s_free_blocks_count; /* Free blocks count */
    __le32  s_free_inodes_count; /* Free inodes count */
```

free inodes and blocks

```
    __le32  s_first_data_block;  /* First Data Block */
    __le32  s_log_block_size;    /* Block size */
```

logarithmic block size:
1024*2^i, i=0,1,2 (1024, 2048, 4096)

```
    __le32  s_log_frag_size;     /* Fragment size */
    __le32  s_blocks_per_group;  /* # Blocks per group */
    __le32  s_frags_per_group;   /* # Fragments per group */
    __le32  s_inodes_per_group;  /* # Inodes per group */
```

inodes / blocks in a block group



# Ext2/Ext3 Superblock (4)

```
__le32  s_mtime;             /* Mount time */
__le32  s_wtime;             /* Write time */
__le16  s_mnt_count;         /* Mount count */
__le16  s_max_mnt_count;     /* Maximal mount count */
```

mount count is increased each time the fs is mounted; when a maximum value is reached, it is time for an *fsck*

```
        /* Maximal mount counts between two filesystem checks */
        #define EXT2_DFL_MAX_MNT_COUNT       20  /* Allow 20 mounts */

__le16  s_magic;             /* Magic signature */

        /* The second extended file system magic number */
        #define EXT2_SUPER_MAGIC 0xEF53

__le16  s_state;             /* File system state */

        /*
         * File system states
         */
        #define EXT2_VALID_FS            0x0001  /* Unmounted cleanly */
        #define EXT2_ERROR_FS            0x0002  /* Errors detected */
```

# Ext2/Ext3 Superblock (5)

```
__le16  s_errors;           /* Behaviour when detecting errors */

        /*Behaviour when detecting errors */
        #define EXT2_ERRORS_CONTINUE     1   /* Continue execution */
        #define EXT2_ERRORS_RO           2   /* Remount fs read-only */
        #define EXT2_ERRORS_PANIC        3   /* Panic */
        #define EXT2_ERRORS_DEFAULT      EXT2_ERRORS_CONTINUE

__le16  s_minor_rev_level;  /* minor revision level */
__le32  s_lastcheck;        /* time of last check */
__le32  s_checkinterval;    /* max. time between checks */
```

lastcheck and checkinterval: besides having a mount count there are also automatic *fsck* checks after a given time interval passed

```
__le32  s_creator_os;       /* OS */
__le32  s_rev_level;        /* Revision level */
__le16  s_def_resuid;       /* Default uid for reserved blocks */
__le16  s_def_resgid;       /* Default gid for reserved blocks */
```

„reserved space“ (5 %) for a privileged user.
default: resuid = resgid = 0 (root/root)

```
#define  EXT2_DEF_RESUID     0
#define  EXT2_DEF_RESGID     0
```



# Ext2/Ext3 Superblock (6)

```
__le32  s_first_ino;            /* First non-reserved inode */
__le16  s_inode_size;           /* size of inode structure */
__le16  s_block_group_nr;       /* block group # of this superblock */
__le32  s_feature_compat;       /* compatible feature set */
__le32  s_feature_incompat;     /* incompatible feature set */
__le32  s_feature_ro_compat;    /* readonly-compatible feature set */
```

three feature categories that the kernel uses for deciding, whether and how it will mount this filesystem:

- compatible: mount filesystem r/w, even if unsupported
- ro-comp.: mount filesystem r/o if unsupported
- incompatible: do not mount filesystem at all if unsupported

`s_feature_compat`, `s_feature_ro_compat`, `s_feature_incompat` are bitmaps which describe the features active *in this filesystem*, e.g.

```
#define EXT2_FEATURE_COMPAT_DIR_PREALLOC  0x0001
#define EXT2_FEATURE_COMPAT_IMAGIC_INODES 0x0002
#define EXT3_FEATURE_COMPAT_HAS_JOURNAL   0x0004
...
#define EXT2_FEATURE_INCOMPAT_COMPRESSION 0x0001
#define EXT2_FEATURE_INCOMPAT_FILETYPE    0x0002
#define EXT3_FEATURE_INCOMPAT_RECOVER     0x0004
...
```



# Ext2/Ext3 Superblock (7)

```
__u8    s_uuid[16];             /* 128-bit uuid for volume */
char    s_volume_name[16];      /* volume name */
char    s_last_mounted[64];     /* directory where last mounted */
__le32  s_algorithm_usage_bitmap; /* For compression */
/*
 * Performance hints.  Directory preallocation should only
 * happen if the EXT2_COMPAT_PREALLOC flag is on.
 */
__u8    s_prealloc_blocks;      /* Nr of blocks to try to preallocate */
__u8    s_prealloc_dir_blocks;  /* Nr to preallocate for dirs */
```

**Pre-allocation:** when allocating a block, Ext2 will not only allocate the requested block for the file, but will reserve several blocks in advance: this reduces potential fragmentation and speeds up further block requests for the same file which follow briefly after the current request (e.g. when sequentially writing to a file)

```
__u16   s_padding1;
```



# Ext2/Ext3 Superblock (8)

```
    /*
     * Journaling support valid if EXT3_FEATURE_COMPAT_HAS_JOURNAL set.
     */
    __u8    s_journal_uuid[16];  /* uuid of journal superblock */
    __u32   s_journal_inum;      /* inode number of journal file */
    __u32   s_journal_dev;       /* device number of journal file */
```

the journal can reside on a different filesystem; this not only a file inode, but also a device number

```
    __u32   s_last_orphan;       /* start of list of inodes to delete */
    __u32   s_hash_seed[4];      /* HTREE hash seed */
    __u8    s_def_hash_version;  /* Default hash version to use */
    __u8    s_reserved_char_pad;
    __u16   s_reserved_word_pad;
    __le32  s_default_mount_opts;
    __le32  s_first_meta_bg;     /* First metablock block group */
    __u32   s_reserved[190];     /* Padding to the end of the block */
};
```

# Ext2 Superblock in Memory (1)

file: *include/linux/ext2_fs_sb.h*

```
/* second extended-fs super-block data in memory */
struct ext2_sb_info {
     unsigned long s_frag_size;      /* Size of a fragment in bytes */
     unsigned long s_frags_per_block;    /* Number of fragments per block */
     unsigned long s_inodes_per_block;   /* Number of inodes per block */
     unsigned long s_frags_per_group;    /* Number of fragments in a group */
     unsigned long s_blocks_per_group;   /* Number of blocks in a group */
     unsigned long s_inodes_per_group;   /* Number of inodes in a group */
     unsigned long s_itb_per_group;  /* Number of inode table blocks per group */
     unsigned long s_gdb_count;      /* Number of group descriptor blocks */
     unsigned long s_desc_per_block; /* Number of group descriptors per block */
     unsigned long s_groups_count;       /* Number of groups in the fs */
     struct buffer_head * s_sbh;     /* Buffer containing the super block */
     struct ext2_super_block * s_es; /* Pointer to the super block in the buffer */
     struct buffer_head ** s_group_desc;
     unsigned long  s_mount_opt;

           #define EXT2_MOUNT_CHECK         0x0001   /* Do mount-time checks */
           #define EXT2_MOUNT_OLDALLOC      0x0002   /* Don't use the new Orlov allocator */
           #define EXT2_MOUNT_GRPID         0x0004   /* Create files with directory's group */
           #define EXT2_MOUNT_DEBUG         0x0008   /* Some debugging messages */
           #define EXT2_MOUNT_ERRORS_CONT   0x0010   /* Continue on errors */
           #define EXT2_MOUNT_ERRORS_RO     0x0020   /* Remount fs ro on errors */
           #define EXT2_MOUNT_ERRORS_PANIC  0x0040   /* Panic on errors */
           #define EXT2_MOUNT_MINIX_DF      0x0080   /* Mimics the Minix statfs */
           #define EXT2_MOUNT_NOBH          0x0100   /* No buffer_heads */
           #define EXT2_MOUNT_NO_UID32      0x0200   /* Disable 32-bit UIDs */
           #define EXT2_MOUNT_XATTR_USER    0x4000   /* Extended user attributes */
           #define EXT2_MOUNT_POSIX_ACL     0x8000   /* POSIX Access Control Lists */
```



# Ext2 Superblock in Memory (2)

```
     uid_t s_resuid;
     gid_t s_resgid;
     unsigned short s_mount_state;
     unsigned short s_pad;
     int s_addr_per_block_bits;
     int s_desc_per_block_bits;
     int s_inode_size;
     int s_first_ino;
     spinlock_t s_next_gen_lock;
     u32 s_next_generation;
     unsigned long s_dir_count;
     u8 *s_debts;
     struct percpu_counter s_freeblocks_counter;
     struct percpu_counter s_freeinodes_counter;
     struct percpu_counter s_dirs_counter;
     struct blockgroup_lock s_blockgroup_lock;
};
```



# Ext2 Inode (1)

file: *include/linux/ext2_fs.h*

```
/* Structure of an inode on the disk */
struct ext2_inode {
        __le16  i_mode;                 /* File mode */
        __le16  i_uid;                  /* Low 16 bits of Owner Uid */
        __le32  i_size;                 /* Size in bytes */
        __le32  i_atime;                /* Access time */
        __le32  i_ctime;                /* Creation time */
        __le32  i_mtime;                /* Modification time */
        __le32  i_dtime;                /* Deletion Time */
        __le16  i_gid;                  /* Low 16 bits of Group Id */
        __le16  i_links_count;          /* Links count */
        __le32  i_blocks;               /* Blocks count */
        __le32  i_flags;                /* File flags */
```

Flags:

```
EXT2_SECRM_FL          0x00000001   secure deletion
EXT2_UNRM_FL           0x00000002   record for undelete
EXT2_COMPR_FL          0x00000004   compressed file
EXT2_SYNC_FL           0x00000008   synchronous updates
EXT2_IMMUTABLE_FL      0x00000010   immutable file
EXT2_APPEND_FL         0x00000020   append only
EXT2_NODUMP_FL         0x00000040   do not dump/delete file
EXT2_NOATIME_FL        0x00000080   do not update .i_atime
EXT2_DIRTY_FL          0x00000100   dirty (file is in use?)
EXT2_COMPRBLK_FL       0x00000200   compressed blocks
EXT2_NOCOMPR_FL        0x00000400   access raw compressed data
EXT2_ECOMPR_FL         0x00000800   compression error
EXT2_BTREE_FL          0x00010000   b-tree format directory
EXT2_INDEX_FL          0x00010000   Hash indexed directory
EXT3_JOURNAL_DATA_FL   0x00040000   journal file data
EXT2_RESERVED_FL       0x80000000   reserved for ext2 implementation
```



# Ext2 Inode (2)

```
        union {
                struct {
                        __le32  l_i_reserved1;
                } linux1;
            [...]
        } osd1;                                 /* OS dependent 1 */
        __le32  i_block[EXT2_N_BLOCKS];         /* Pointers to blocks */
        __le32  i_generation;                   /* File version (for NFS) */
        __le32  i_file_acl;                     /* File ACL */
        __le32  i_dir_acl;                      /* Directory ACL */

             #define i_size_high    i_dir_acl
```

for large files with a 64 bit length, i_dir_acl (which is otherwise unused for files) is used as upper 32 bits, i_size lower 32 bits

```
        __le32  i_faddr;                        /* Fragment address */
        union {
                struct {
                        __u8    l_i_frag;       /* Fragment number */
                        __u8    l_i_fsize;      /* Fragment size */
                        __u16   i_pad1;
                        __le16  l_i_uid_high;   /* these 2 fields    */
                        __le16  l_i_gid_high;   /* were reserved2[0] */
                        __u32   l_i_reserved2;
                } linux2;
            [...]
        } osd2;                                 /* OS dependent 2 */
};
```

# Extra Flags

- Ext2/Ext3 extra flags (immutable, append-only etc.) can be modified with **chattr**
  (cf. *9. File Systems (2)/Folie 21*)

```
NAME
       chattr - change file attributes on a Linux second extended file system

SYNOPSIS
       chattr [ -RV ] [ -v version ] [ mode ] files...

DESCRIPTION
       chattr changes the file attributes on a Linux second extended file system.

       The format of a symbolic mode is +-=[ASacDdIijsTtu].

       The operator `+' causes the selected attributes to be added to the existing
       attributes of the files; `-' causes them to be removed; and `=' causes them
       to be the only attributes that the files have.
```



# Extended Attributes (1)

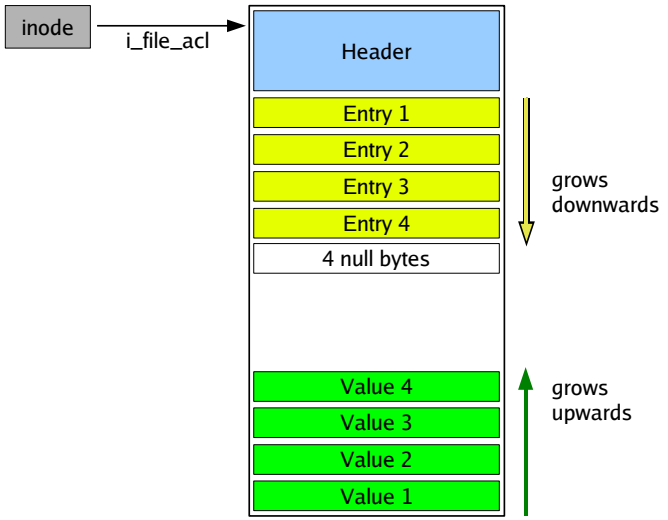
- inode size: 128 bytes
  - no space for extended attributes
  - increasing inodes to 256 bytes is inefficient
- solution: separate block for extended attributes (`ext2_xattr_entry`)
  - xattr header:

file: *fs/ext2/xattr.h*

```
struct ext2_xattr_header {
    __le32  h_magic;        /* magic number for identification */
    __le32  h_refcount;     /* reference count */
    __le32  h_blocks;       /* number of disk blocks used */
    __le32  h_hash;         /* hash value of all attributes */
    __u32   h_reserved[4];  /* zero right now */
};
```



# Extended Attributes (2)



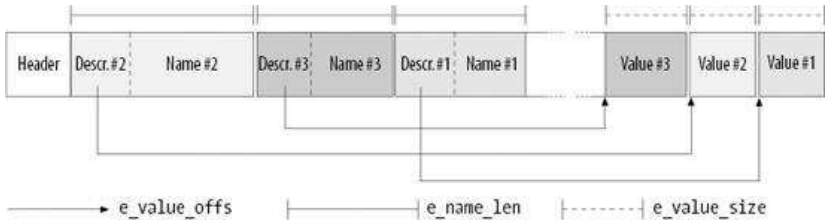



# Extended Attributes (3)

xattr entry:

file: *fs/ext2/xattr.h*

```
struct ext2_xattr_entry {
    __u8    e_name_len;     /* length of name */
    __u8    e_name_index;   /* attribute name index */
    __le16  e_value_offs;   /* offset in disk block of value */
    __le32  e_value_block;  /* disk block attribute is stored on (n/i) */
    __le32  e_value_size;   /* size of attribute value */
    __le32  e_hash;         /* hash value of name and value */
    char    e_name[0];      /* attribute name */
};
```




Picture: Understanding the Linux Kernel (Bovet/Cesati)

# Extended Attributes (4)

- modify and query with **setfattr**, **getfattr**, **attr**:

```
amd64:/home/esser # setfattr -n user.foo -v betriebssysteme test.txt
amd64:/home/esser # getfattr -d test.txt
# file: test.txt
user.foo="betriebssysteme"

amd64:/home/esser # attr -g user.foo test.txt
Attribute "user.foo" had a 5 byte value for test.txt:
hallo
```



# Three Kinds of Attributes

**don't confuse:**

- standard Unix file attributes
  - UID, GID
  - standard access rights rwx,
  - access times, ...
- extra flags
  - immutable, compressed, secure deletion, ...
- extended attributes
  - arbitrary, freely-definable attributes (incl. ACLs)