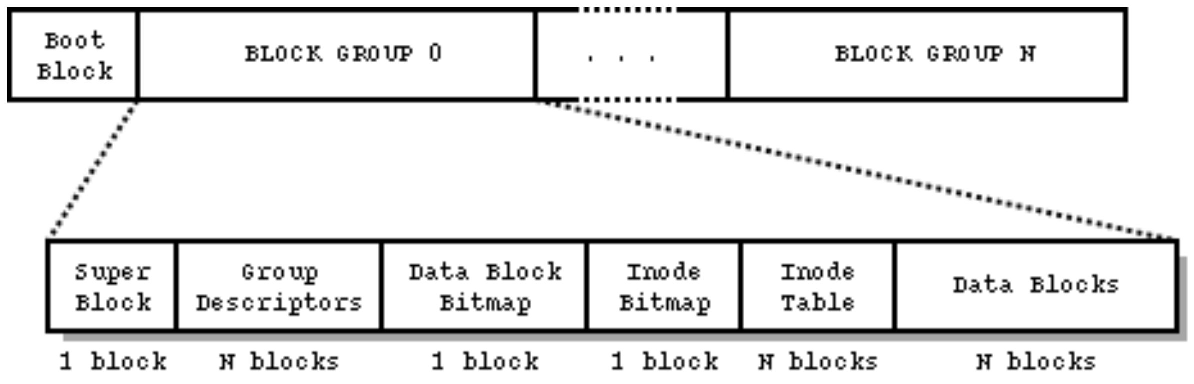
ext4 File System

Trevor Bakker

The University of Texas at Arlington

Structure of an Ext4 Filesystem



- The first 1024 bytes of the disk, the "boot block", are reserved for the partition boot sectors and are unused by the Ext4 filesystem. The rest of the partition is split into block groups,

The SuperBlock

It contains information such as the total number of blocks on disk, the size of a block (usually 4096 bytes), the number of free blocks, etc.

Part of this structure:

```
struct ext2_super_block {
   __u32 s_inodes_count;    /* Inodes count */
   __u32 s_blocks_count;    /* Blocks count */
   ...
   __u32 s_free_blocks_count;  /* Free blocks count */
   __u32 s_free_inodes_count;  /* Free inodes count */
   __u32 s_first_data_block;   /* First Data Block */
   __u32 s_log_block_size;  /* Block size */
   ...
   __u32 s_blocks_per_group;   /* # Blocks per group */
   ...
   __u16 s_magic;   /* Magic signature */
   ...
```

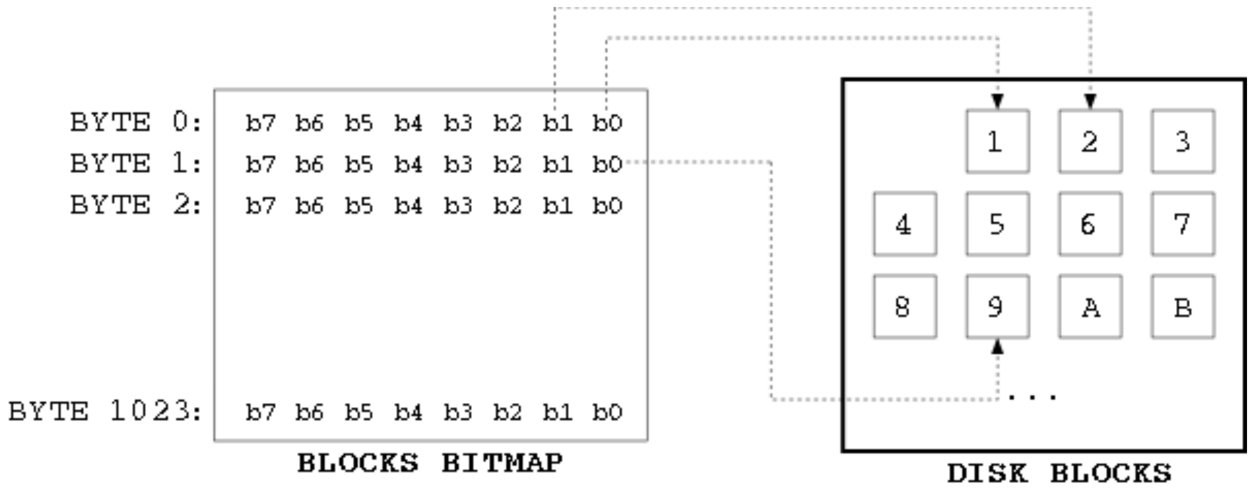
Group Descriptors

- In the blocks immediately following the super-block reside the list of block-group descriptors. This list contains a descriptor for each block group on the disk. In the case of a floppy, there is only one block group and therefore one group descriptor.

```
struct ext2_group_desc
{
   __u32 bg_block_bitmap;  /* Blocks bitmap block */
   __u32 bg_inode_bitmap;  /* Inodes bitmap block */
   __u32 bg_inode_table;      /* Inodes table block */
   __u16 bg_free_blocks_count;  /* Free blocks count */
   __u16 bg_free_inodes_count;  /* Free inodes count */
   __u16 bg_used_dirs_count; /* Directories count */
   __u16 bg_pad;
   __u32 bg_reserved[3];
};
```

The blocks and inodes bitmaps

- Each bit represents a specific block (blocks bitmap) or inode (inode bitmap) in the block group. A bit value of 0 indicates that the block/inode is free, while a value of 1 indicates that the block/inode is being used. A bitmap always refers to the block-group it belongs to, and its size must fit in one block.



The inode table

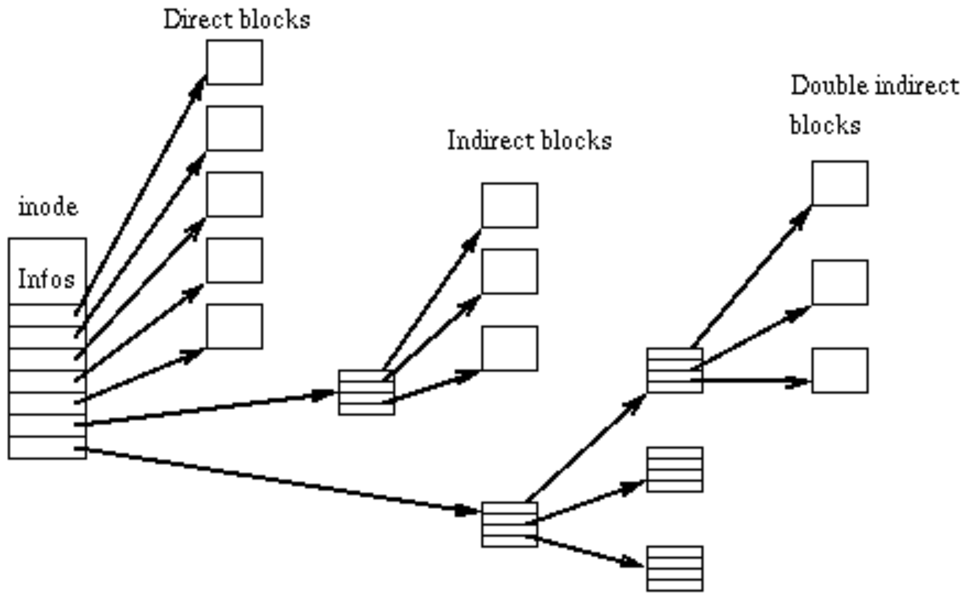
- The inode table consists of a series of consecutive blocks, each of which contains a predefined number of inodes.

- The inode table contains everything the operating system needs to know about a file, including the type of file, permissions, owner, and, most important, where its data blocks are located on disk. It is no surprise therefore that this table needs to be accessed very frequently and its read access time should be minimized as much as possible.

The inode table

```
struct ext2_inode {
        __u16   i_mode;         /* File type and access rights */
        __u16   i_uid;          /* Low 16 bits of Owner Uid */
        __u32   i_size;         /* Size in bytes */
        __u32   i_atime;        /* Access time */
        __u32   i_ctime;        /* Creation time */
        __u32   i_mtime;        /* Modification time */
        __u32   i_dtime;        /* Deletion Time */
        __u16   i_gid;          /* Low 16 bits of Group Id */
        __u16   i_links_count;  /* Links count */
        __u32   i_blocks;       /* Blocks count */
        __u32   i_flags;        /* File flags */
   ...
   __u32   i_block[EXT2_N_BLOCKS];  /* Pointers to blocks */
   ...
};
```

The inode structure

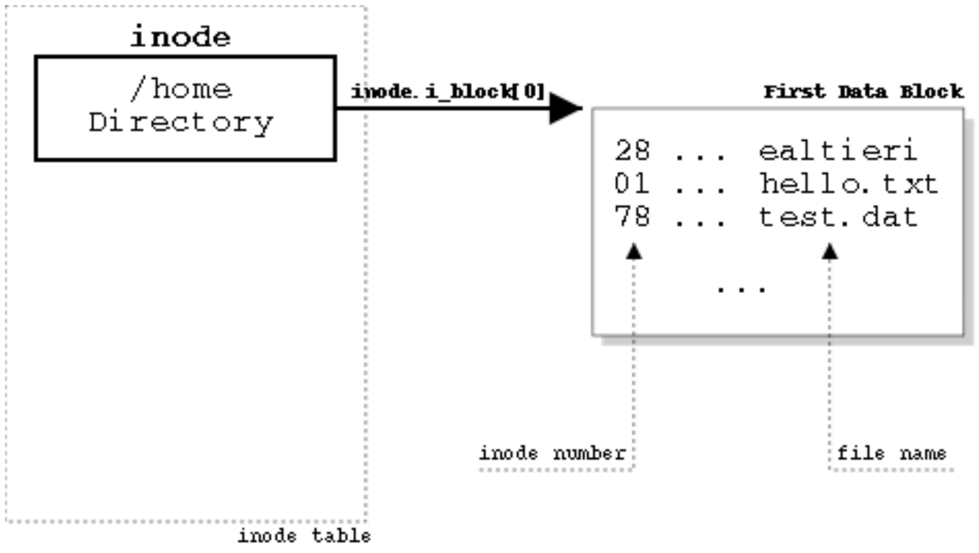


Quote from the Linux kernel documentation for ext2:
"There are pointers to the first 12 blocks which contain the file's data in the inode. There is a pointer to an indirect block (which contains pointers to the next set of blocks), a pointer to a doubly indirect block and a pointer to a trebly indirect block."

So, there is a structure in ext2 that has 15 pointers. Pointers 1 to 12 point to direct blocks, pointer 13 points to an indirect block, pointer 14 points to a doubly indirect block, and pointer 15 points to a trebly indirect block.

Directory entries in the inode table

- In the case of directory entries, the data blocks pointed by i_block[] contain a list of the files in the directory and their respective inode numbers.
- Finding: /home/ealtieri/hello.txt



```
struct ext2_dir_entry_2 {
     __u32     inode;              /* Inode number */
     __u16     rec_len;       /* Directory entry length */
     __u8 name_len;      /* Name length */
     __u8 file_type;
     char name[EXT2_NAME_LEN];    /* File name */
};
```