Super Block

The superblock records various information about the enclosing filesystem, such as block counts, inode counts, supported features, maintenance information, and more.

If the sparse_super feature flag is set, redundant copies of the superblock and group descriptors are kept only in the groups whose group number is either 0 or a power of 3, 5, or 7. If the flag is not set, redundant copies are kept in all groups.

The superblock checksum is calculated against the superblock structure, which includes the FS UUID.

The ext4 superblock is laid out as follows in `struct ext4_super_block`:

Offset	**Size**	**Name**	**Description**
0x0	__le32	s_inodes_count	Total inode count.
0x4	__le32	s_blocks_count_lo	Total block count.
0x8	__le32	s_r_blocks_count_lo	This number of blocks can only be allocated by the super-user.
0xC	__le32	s_free_blocks_count_lo	Free block count.
0x10	__le32	s_free_inodes_count	Free inode count.
0x14	__le32	s_first_data_block	First data block. This must be at least 1 for 1k-block filesystems and is typically 0 for all other block sizes.
0x18	__le32	s_log_block_size	Block size is 2 ^ (10 + s_log_block_size).
0x1C	__le32	s_log_cluster_size	Cluster size is 2 ^ (10 + s_log_cluster_size) blocks if bigalloc is enabled. Otherwise s_log_cluster_size must equal s_log_block_size.
0x20	__le32	s_blocks_per_group	Blocks per group.
0x24	__le32	s_clusters_per_group	Clusters per group, if bigalloc is enabled. Otherwise s_clusters_per_group must equal s_blocks_per_group.
0x28	__le32	s_inodes_per_group	Inodes per group.
0x2C	__le32	s_mtime	Mount time, in seconds since the epoch.
0x30	__le32	s_wtime	Write time, in seconds since the epoch.
0x34	__le16	s_mnt_count	Number of mounts since the last fsck.
0x36	__le16	s_max_mnt_count	Number of mounts beyond which a fsck is needed.
0x38	__le16	s_magic	Magic signature, 0xEF53
0x3A	__le16	s_state	File system state. See super_state for more info.
0x3C	__le16	s_errors	Behaviour when detecting errors. See super_errors for more info.
0x3E	__le16	s_minor_rev_level	Minor revision level.
0x40	__le32	s_lastcheck	Time of last check, in seconds since the epoch.
0x44	__le32	s_checkinterval	Maximum time between checks, in seconds.
0x48	__le32	s_creator_os	Creator OS. See the table super_creator for more info.
0x4C	__le32	s_rev_level	Revision level. See the table super_revision for more info.
0x50	__le16	s_def_resuid	Default uid for reserved blocks.
0x52	__le16	s_def_resgid	Default gid for reserved blocks.
			These fields are for EXT4_DYNAMIC_REV superblocks only. Note: the difference between the compatible feature set and the incompatible feature set is that if there is a bit set in the incompatible feature set that the kernel doesn't know about, it should refuse to mount the filesystem. e2fsck's requirements are more strict; if it doesn't know about a feature in either the compatible or incompatible feature set, it must abort and not try to meddle with things it doesn't understand...
0x54	__le32	s_first_ino	First non-reserved inode.
0x58	__le16	s_inode_size	Size of inode structure, in bytes.
0x5A	__le16	s_block_group_nr	Block group # of this superblock.
0x5C	__le32	s_feature_compat	Compatible feature set flags. Kernel can still read/write this fs even if it doesn't understand a flag; fsck should not do that. See the super_compat table for more info.
0x60	__le32	s_feature_incompat	Incompatible feature set. If the kernel or fsck doesn't understand one of these bits, it should stop. See the super_incompat table for more info.

Offset	Size	Name	Description
0x64	__le32	s_feature_ro_compat	Readonly-compatible feature set. If the kernel doesn't understand one of these bits, it can still mount read-only. See the super_rocompat table for more info.
0x68	__u8	s_uuid[16]	128-bit UUID for volume.
0x78	char	s_volume_name[16]	Volume label.
0x88	char	s_last_mounted[64]	Directory where filesystem was last mounted.
0xC8	__le32	s_algorithm_usage_bitmap	For compression (Not used in e2fsprogs/Linux)
			Performance hints. Directory preallocation should only happen if the EXT4_FEATURE_COMPAT_DIR_PREALLOC flag is on.
0xCC	__u8	s_prealloc_blocks	#. of blocks to try to preallocate for ... files? (Not used in e2fsprogs/Linux)
0xCD	__u8	s_prealloc_dir_blocks	#. of blocks to preallocate for directories. (Not used in e2fsprogs/Linux)
0xCE	__le16	s_reserved_gdt_blocks	Number of reserved GDT entries for future filesystem expansion.
			Journalling support is valid only if EXT4_FEATURE_COMPAT_HAS_JOURNAL is set.
0xD0	__u8	s_journal_uuid[16]	UUID of journal superblock
0xE0	__le32	s_journal_inum	inode number of journal file.
0xE4	__le32	s_journal_dev	Device number of journal file, if the external journal feature flag is set.
0xE8	__le32	s_last_orphan	Start of list of orphaned inodes to delete.
0xEC	__le32	s_hash_seed[4]	HTREE hash seed.
0xFC	__u8	s_def_hash_version	Default hash algorithm to use for directory hashes. See super_def_hash for more info.
0xFD	__u8	s_jnl_backup_type	If this value is 0 or EXT3_JNL_BACKUP_BLOCKS (1), then the `s_jnl_blocks` field contains a duplicate copy of the inode's `i_block[]` array and `i_size`.
0xFE	__le16	s_desc_size	Size of group descriptors, in bytes, if the 64bit incompat feature flag is set.
0x100	__le32	s_default_mount_opts	Default mount options. See the super_mountopts table for more info.
0x104	__le32	s_first_meta_bg	First metablock block group, if the meta_bg feature is enabled.
0x108	__le32	s_mkfs_time	When the filesystem was created, in seconds since the epoch.
0x10C	__le32	s_jnl_blocks[17]	Backup copy of the journal inode's `i_block[]` array in the first 15 elements and i_size_high and i_size in the 16th and 17th elements, respectively.
			64bit support is valid only if EXT4_FEATURE_COMPAT_64BIT is set.
0x150	__le32	s_blocks_count_hi	High 32-bits of the block count.
0x154	__le32	s_r_blocks_count_hi	High 32-bits of the reserved block count.
0x158	__le32	s_free_blocks_count_hi	High 32-bits of the free block count.
0x15C	__le16	s_min_extra_isize	All inodes have at least # bytes.
0x15E	__le16	s_want_extra_isize	New inodes should reserve # bytes.
0x160	__le32	s_flags	Miscellaneous flags. See the super_flags table for more info.
0x164	__le16	s_raid_stride	RAID stride. This is the number of logical blocks read from or written to the disk before moving to the next disk. This affects the placement of filesystem metadata, which will hopefully make RAID storage faster.
0x166	__le16	s_mmp_interval	#. seconds to wait in multi-mount prevention (MMP) checking. In theory, MMP is a mechanism to record in the superblock which host and device have mounted the filesystem, in order to prevent multiple mounts. This feature does not seem to be implemented...
0x168	__le64	s_mmp_block	Block # for multi-mount protection data.
0x170	__le32	s_raid_stripe_width	RAID stripe width. This is the number of logical blocks read from or written to the disk before coming back to the current disk. This is used by the block allocator to try to reduce the number of read-modify-write operations in a RAID5/6.
0x174	__u8	s_log_groups_per_flex	Size of a flexible block group is 2 ^ `s_log_groups_per_flex`.
0x175	__u8	s_checksum_type	Metadata checksum algorithm type. The only valid value is 1 (crc32c).

Offset	Size	Name	Description
0x176	__le16	s_reserved_pad	
0x178	__le64	s_kbytes_written	Number of KiB written to this filesystem over its lifetime.
0x180	__le32	s_snapshot_inum	inode number of active snapshot. (Not used in e2fsprogs/Linux.)
0x184	__le32	s_snapshot_id	Sequential ID of active snapshot. (Not used in e2fsprogs/Linux.)
0x188	__le64	s_snapshot_r_blocks_count	Number of blocks reserved for active snapshot's future use. (Not used in e2fsprogs/Linux.)
0x190	__le32	s_snapshot_list	inode number of the head of the on-disk snapshot list. (Not used in e2fsprogs/Linux.)
0x194	__le32	s_error_count	Number of errors seen.
0x198	__le32	s_first_error_time	First time an error happened, in seconds since the epoch.
0x19C	__le32	s_first_error_ino	inode involved in first error.
0x1A0	__le64	s_first_error_block	Number of block involved of first error.
0x1A8	__u8	s_first_error_func[32]	Name of function where the error happened.
0x1C8	__le32	s_first_error_line	Line number where error happened.
0x1CC	__le32	s_last_error_time	Time of most recent error, in seconds since the epoch.
0x1D0	__le32	s_last_error_ino	inode involved in most recent error.
0x1D4	__le32	s_last_error_line	Line number where most recent error happened.
0x1D8	__le64	s_last_error_block	Number of block involved in most recent error.
0x1E0	__u8	s_last_error_func[32]	Name of function where the most recent error happened.
0x200	__u8	s_mount_opts[64]	ASCIIZ string of mount options.
0x240	__le32	s_usr_quota_inum	Inode number of user quota file.
0x244	__le32	s_grp_quota_inum	Inode number of group quota file.
0x248	__le32	s_overhead_blocks	Overhead blocks/clusters in fs. (Huh? This field is always zero, which means that the kernel calculates it dynamically.)
0x24C	__le32	s_backup_bgs[2]	Block groups containing superblock backups (if sparse_super2)
0x254	__u8	s_encrypt_algos[4]	Encryption algorithms in use. There can be up to four algorithms in use at any time; valid algorithm codes are given in the super_encrypt table below.
0x258	__u8	s_encrypt_pw_salt[16]	Salt for the string2key algorithm for encryption.
0x268	__le32	s_lpf_ino	Inode number of lost+found
0x26C	__le32	s_prj_quota_inum	Inode that tracks project quotas.
0x270	__le32	s_checksum_seed	Checksum seed used for metadata_csum calculations. This value is crc32c(~0, $orig_fs_uuid).
0x274	__u8	s_wtime_hi	Upper 8 bits of the s_wtime field.
0x275	__u8	s_mtime_hi	Upper 8 bits of the s_mtime field.
0x276	__u8	s_mkfs_time_hi	Upper 8 bits of the s_mkfs_time field.
0x277	__u8	s_lastcheck_hi	Upper 8 bits of the s_lastcheck_hi field.
0x278	__u8	s_first_error_time_hi	Upper 8 bits of the s_first_error_time_hi field.
0x279	__u8	s_last_error_time_hi	Upper 8 bits of the s_last_error_time_hi field.
0x27A	__u8	s_pad[2]	Zero padding.
0x27C	__le16	s_encoding	Filename charset encoding.
0x27E	__le16	s_encoding_flags	Filename charset encoding flags.
0x280	__le32	s_orphan_file_inum	Orphan file inode number.
0x284	__le32	s_reserved[94]	Padding to the end of the block.
0x3FC	__le32	s_checksum	Superblock checksum.

The superblock state is some combination of the following:

Value	Description
0x0001	Cleanly umounted
0x0002	Errors detected
0x0004	Orphans being recovered

The superblock error policy is one of the following:

Value	Description
1	Continue
2	Remount read-only
3	Panic

The filesystem creator is one of the following:

Value	Description
0	Linux

Value	Description
1	Hurd
2	Masix
3	FreeBSD
4	Lites

The superblock revision is one of the following:

Value	Description
0	Original format
1	v2 format w/ dynamic inode sizes

Note that `EXT4_DYNAMIC_REV` refers to a revision 1 or newer filesystem.

The superblock compatible features field is a combination of any of the following:

Value	Description
0x1	Directory preallocation (COMPAT_DIR_PREALLOC).
0x2	"imagic inodes". Not clear from the code what this does (COMPAT_IMAGIC_INODES).
0x4	Has a journal (COMPAT_HAS_JOURNAL).
0x8	Supports extended attributes (COMPAT_EXT_ATTR).
0x10	Has reserved GDT blocks for filesystem expansion (COMPAT_RESIZE_INODE). Requires RO_COMPAT_SPARSE_SUPER.
0x20	Has directory indices (COMPAT_DIR_INDEX).
0x40	"Lazy BG". Not in Linux kernel, seems to have been for uninitialized block groups? (COMPAT_LAZY_BG)
0x80	"Exclude inode". Not used. (COMPAT_EXCLUDE_INODE).
0x100	"Exclude bitmap". Seems to be used to indicate the presence of snapshot-related exclude bitmaps? Not defined in kernel or used in e2fsprogs (COMPAT_EXCLUDE_BITMAP).
0x200	Sparse Super Block, v2. If this flag is set, the SB field s_backup_bgs points to the two block groups that contain backup superblocks (COMPAT_SPARSE_SUPER2).
0x400	Fast commits supported. Although fast commits blocks are backward incompatible, fast commit blocks are not always present in the journal. If fast commit blocks are present in the journal, JBD2 incompat feature (JBD2_FEATURE_INCOMPAT_FAST_COMMIT) gets set (COMPAT_FAST_COMMIT).
0x1000	Orphan file allocated. This is the special file for more efficient tracking of unlinked but still open inodes. When there may be any entries in the file, we additionally set proper rocompat feature (RO_COMPAT_ORPHAN_PRESENT).

The superblock incompatible features field is a combination of any of the following:

Value	Description
0x1	Compression (INCOMPAT_COMPRESSION).
0x2	Directory entries record the file type. See ext4_dir_entry_2 below (INCOMPAT_FILETYPE).
0x4	Filesystem needs recovery (INCOMPAT_RECOVER).
0x8	Filesystem has a separate journal device (INCOMPAT_JOURNAL_DEV).
0x10	Meta block groups. See the earlier discussion of this feature (INCOMPAT_META_BG).
0x40	Files in this filesystem use extents (INCOMPAT_EXTENTS).
0x80	Enable a filesystem size of 2^64 blocks (INCOMPAT_64BIT).
0x100	Multiple mount protection (INCOMPAT_MMP).
0x200	Flexible block groups. See the earlier discussion of this feature (INCOMPAT_FLEX_BG).
0x400	Inodes can be used to store large extended attribute values (INCOMPAT_EA_INODE).
0x1000	Data in directory entry (INCOMPAT_DIRDATA). (Not implemented?)
0x2000	Metadata checksum seed is stored in the superblock. This feature enables the administrator to change the UUID of a metadata_csum filesystem while the filesystem is mounted; without it, the checksum definition requires all metadata blocks to be rewritten (INCOMPAT_CSUM_SEED).
0x4000	Large directory >2GB or 3-level htree (INCOMPAT_LARGEDIR). Prior to this feature, directories could not be larger than 4GiB and could not have an htree more than 2 levels deep. If this feature is enabled, directories can be larger than 4GiB and have a maximum htree depth of 3.
0x8000	Data in inode (INCOMPAT_INLINE_DATA).
0x10000	Encrypted inodes are present on the filesystem. (INCOMPAT_ENCRYPT).

The superblock read-only compatible features field is a combination of any of the following:

Value	Description
0x1	Sparse superblocks. See the earlier discussion of this feature (RO_COMPAT_SPARSE_SUPER).
0x2	This filesystem has been used to store a file greater than 2GiB (RO_COMPAT_LARGE_FILE).

Value	Description
0x4	Not used in kernel or e2fsprogs (RO_COMPAT_BTREE_DIR).
0x8	This filesystem has files whose sizes are represented in units of logical blocks, not 512-byte sectors. This implies a very large file indeed! (RO_COMPAT_HUGE_FILE)
0x10	Group descriptors have checksums. In addition to detecting corruption, this is useful for lazy formatting with uninitialized groups (RO_COMPAT_GDT_CSUM).
0x20	Indicates that the old ext3 32,000 subdirectory limit no longer applies (RO_COMPAT_DIR_NLINK). A directory's i_links_count will be set to 1 if it is incremented past 64,999.
0x40	Indicates that large inodes exist on this filesystem (RO_COMPAT_EXTRA_ISIZE).
0x80	This filesystem has a snapshot (RO_COMPAT_HAS_SNAPSHOT).
0x100	Quota (RO_COMPAT_QUOTA).
0x200	This filesystem supports “bigalloc”, which means that file extents are tracked in units of clusters (of blocks) instead of blocks (RO_COMPAT_BIGALLOC).
0x400	This filesystem supports metadata checksumming. (RO_COMPAT_METADATA_CSUM; implies RO_COMPAT_GDT_CSUM, though GDT_CSUM must not be set)
0x800	Filesystem supports replicas. This feature is neither in the kernel nor e2fsprogs. (RO_COMPAT_REPLICA)
0x1000	Read-only filesystem image; the kernel will not mount this image read-write and most tools will refuse to write to the image. (RO_COMPAT_READONLY)
0x2000	Filesystem tracks project quotas. (RO_COMPAT_PROJECT)
0x8000	Verity inodes may be present on the filesystem. (RO_COMPAT_VERITY)
0x10000	Indicates orphan file may have valid orphan entries and thus we need to clean them up when mounting the filesystem (RO_COMPAT_ORPHAN_PRESENT).

The `s_def_hash_version` field is one of the following:

Value	Description
0x0	Legacy.
0x1	Half MD4.
0x2	Tea.
0x3	Legacy, unsigned.
0x4	Half MD4, unsigned.
0x5	Tea, unsigned.

The `s_default_mount_opts` field is any combination of the following:

Value	Description
0x0001	Print debugging info upon (re)mount. (EXT4_DEFM_DEBUG)
0x0002	New files take the gid of the containing directory (instead of the fsgid of the current process). (EXT4_DEFM_BSDGROUPS)
0x0004	Support userspace-provided extended attributes. (EXT4_DEFM_XATTR_USER)
0x0008	Support POSIX access control lists (ACLs). (EXT4_DEFM_ACL)
0x0010	Do not support 32-bit UIDs. (EXT4_DEFM_UID16)
0x0020	All data and metadata are commited to the journal. (EXT4_DEFM_JMODE_DATA)
0x0040	All data are flushed to the disk before metadata are committed to the journal. (EXT4_DEFM_JMODE_ORDERED)
0x0060	Data ordering is not preserved; data may be written after the metadata has been written. (EXT4_DEFM_JMODE_WBACK)
0x0100	Disable write flushes. (EXT4_DEFM_NOBARRIER)
0x0200	Track which blocks in a filesystem are metadata and therefore should not be used as data blocks. This option will be enabled by default on 3.18, hopefully. (EXT4_DEFM_BLOCK_VALIDITY)
0x0400	Enable DISCARD support, where the storage device is told about blocks becoming unused. (EXT4_DEFM_DISCARD)
0x0800	Disable delayed allocation. (EXT4_DEFM_NODELALLOC)

The `s_flags` field is any combination of the following:

Value	Description
0x0001	Signed directory hash in use.
0x0002	Unsigned directory hash in use.
0x0004	To test development code.

The `s_encrypt_algos` list can contain any of the following:

Value	Description
0	Invalid algorithm (ENCRYPTION_MODE_INVALID).
1	256-bit AES in XTS mode (ENCRYPTION_MODE_AES_256_XTS).

Value	Description
2	256-bit AES in GCM mode (ENCRYPTION_MODE_AES_256_GCM).
3	256-bit AES in CBC mode (ENCRYPTION_MODE_AES_256_CBC).

Total size of the superblock is 1024 bytes.