

# Btrfs

**Btrfs** (pronounced as "butter fuss",<sup>[11]</sup> "better F S",<sup>[8]</sup> "butter F S",<sup>[12]</sup> "b-tree F S",<sup>[12]</sup> or simply by spelling it out) is a computer storage format that combines a file system based on the copy-on-write (COW) principle with a logical volume manager (not to be confused with Linux's LVM), developed together. It was initially designed at Oracle Corporation in 2007 for use in Linux, and since November 2013, the file system's on-disk format has been declared stable in the Linux kernel.<sup>[13]</sup> According to Oracle, Btrfs "is not a true acronym".<sup>[14]</sup>

Btrfs is intended to address the lack of pooling, snapshots, checksums, and integral multi-device spanning in Linux file systems.<sup>[8]</sup> Chris Mason, the principal Btrfs author, stated that its goal was "to let [Linux] scale for the storage that will be available. Scaling is not just about addressing the storage but also means being able to administer and to manage it with a clean interface that lets people see what's being used and makes it more reliable".<sup>[15]</sup>

Contents
<b>History</b>
<b>Features</b>
<ul style="list-style-type: none"><li><u>Implemented</u></li><li><u>Implemented but not recommended for production use</u></li><li><u>Planned but not yet implemented</u></li><li><u>Cloning</u></li><li><u>Subvolumes and snapshots</u></li><li><u>Send–receive</u></li><li><u>Quota groups</u></li><li><u>In-place conversion from ext2/3/4 and ReiserFS</u></li><li><u>Union mounting / seed devices</u></li><li><u>Encryption</u></li><li><u>Checking and recovery</u></li></ul>
<b>Design</b>
<ul style="list-style-type: none"><li><u>File system tree</u></li><li><u>Extents</u></li><li><u>Extent allocation tree</u></li><li><u>Checksum tree and scrubbing</u></li><li><u>Log tree</u></li><li><u>Chunk and device trees</u></li><li><u>Relocation trees</u></li></ul>

Btrfs	
<b>Developer(s)</b>	<u>Facebook</u> , <u>Fujitsu</u> , <u>Fusion-IO</u> , <u>Intel</u> , <u>Linux Foundation</u> , <u>Netgear</u> , <u>Oracle Corporation</u> , <u>Red Hat</u> , <u>STRATO AG</u> , and <u>openSUSE</u> <sup>[1]</sup>
<b>Full name</b>	B-tree file system
<b>Introduced</b>	Linux kernel 2.6.29, March 2009
Structures	
<b>Directory contents</b>	<u>B-tree</u>
<b>File allocation</b>	<u>Extents</u>
Limits	
<b>Max. volume size</b>	16 EiB <sup>[2][a]</sup>
<b>Max. file size</b>	16 EiB <sup>[2][a]</sup>
<b>Max. number of files</b>	2 <sup>64</sup> <sup>[b][3]</sup>
<b>Max. filename length</b>	255 <u>ASCII</u> characters (fewer for multibyte character <u>encodings</u> such as <u>Unicode</u> )
<b>Allowed characters in filenames</b>	All except ' / ' and NUL (' <span>\0</span> ')
Features	
<b>Dates recorded</b>	Creation (otime), <sup>[4]</sup> modification (mtime), attribute modification

Superblock

**Commercial support**

Supported

No longer supported

**See also**

**Notes**

**References**

**External links**

## History

The core data structure of Btrfs—the copy-on-write B-tree—was originally proposed by IBM researcher Ohad Rodeh at a presentation at USENIX 2007.<sup>[16]</sup> Chris Mason, an engineer working on ReiserFS for SUSE at the time, joined Oracle later that year and began work on a new file system based on these B-trees.<sup>[17]</sup>

In 2008, the principal developer of the ext3 and ext4 file systems, Theodore Ts'o, stated that although ext4 has improved features, it is not a major advance; it uses old technology and is a stop-gap. Ts'o said that Btrfs is the better direction because "it offers improvements in scalability, reliability, and ease of management".<sup>[18]</sup> Btrfs also has "a number of the same design ideas that reiser3/4 had".<sup>[19]</sup>

Btrfs 1.0, with finalized on-disk format, was originally slated for a late-2008 release,<sup>[20]</sup> and was finally accepted into the Linux kernel mainline in 2009.<sup>[21]</sup> Several Linux distributions began offering Btrfs as an experimental choice of root file system during installation.<sup>[22][23][24]</sup>

In July 2011, Btrfs automatic defragmentation and scrubbing features were merged into version 3.0 of the Linux kernel mainline.<sup>[25]</sup> Besides Mason at Oracle, Miao Xie at Fujitsu contributed performance improvements.<sup>[26]</sup> In June 2012, Chris Mason left Oracle for Fusion-io, which he left a year later with Josef Bacik to join Facebook. While at both companies, Mason continued his work on Btrfs.<sup>[27][17]</sup>


In 2012, two Linux distributions moved Btrfs from experimental to production or supported status: Oracle Linux in March,<sup>[28]</sup> followed by SUSE Linux Enterprise in August.<sup>[29]</sup>

In 2015, Btrfs was adopted as the default filesystem for SUSE Linux Enterprise Server 12.<sup>[30]</sup>

In August 2017, Red Hat announced in the release notes for Red Hat Enterprise Linux (RHEL) 7.4 that it no longer planned to move Btrfs, which had been included as a "technology preview" since RHEL 6 beta, to a fully supported feature, noting that it would remain available in the RHEL 7 release series.<sup>[31]</sup> Btrfs was removed from RHEL 8 in May 2019.<sup>[32]</sup>

In 2020, Btrfs was selected as the default file system for Fedora 33.<sup>[33]</sup>

## Features

	(ctime), and access (atime)
<b>Date range</b>	64-bit signed int offset from 1970-01-01T00:00:00Z <sup>[5]</sup>
<b>Date resolution</b>	Nanosecond
<b>Attributes</b>	<u>POSIX</u> and <u>extended attributes</u>
<b>File system permissions</b>	<u>POSIX</u> and <u>ACL</u>
<b>Transparent compression</b>	Yes ( <u>zlib</u> , <u>LZO</u> <sup>[6]</sup> and (since 4.14) <u>ZSTD</u> <sup>[7]</sup> )
<b>Transparent encryption</b>	Planned <sup>[8]</sup>
<b>Data deduplication</b>	Yes <sup>[9]</sup>
<b>Copy-on-write</b>	Yes
<b>Other</b>	
<b>Supported operating systems</b>	<u>Linux</u> , <u>ReactOS</u> <sup>[10]</sup>
<b>Website</b>	<u>btrfs.wiki.kernel.org</u> ( <u>https://btrfs.wiki.kernel.org/</u> ) 

## Implemented

As of version 5.0 of the Linux kernel, Btrfs implements the following features:<sup>[34][35]</sup>

- Mostly self-healing in some configurations due to the nature of copy-on-write
- Online defragmentation and an *autodefrag* mount option<sup>[25]</sup>
- Online volume growth and shrinking
- Online block device addition and removal
- Online balancing (movement of objects between block devices to balance load)
- Offline filesystem check<sup>[36]</sup>
- Online data scrubbing for finding errors and automatically fixing them for files with redundant copies
- RAID 0, RAID 1, and RAID 10<sup>[37]</sup>
- Subvolumes (one or more separately mountable filesystem roots within each disk partition)
- Transparent compression via zlib, LZO<sup>[6]</sup> and (since 4.14) ZSTD,<sup>[7]</sup> configurable per file or volume<sup>[38][39]</sup>
- Atomic writable (via copy-on-write) or read-only<sup>[40]</sup> Snapshots of subvolumes
- File cloning (reflink, copy-on-write) via `cp --reflink <source file> <destination file>`<sup>[41]</sup>
- Checksums on data and metadata (CRC-32C<sup>[42]</sup>). New hash functions are implemented since 5.5:<sup>[43]</sup> xxHash, SHA256, BLAKE2B.
- In-place conversion from ext3/4 to Btrfs (with rollback). This feature regressed around btrfs-progs version 4.0, rewritten from scratch in 4.6.<sup>[44]</sup>
- Union mounting of read-only storage, known as file system seeding (read-only storage used as a copy-on-write backing for a writable Btrfs)<sup>[45]</sup>
- Block discard (reclaims space on some virtualized setups and improves wear leveling on SSDs with TRIM)
- Send/receive (saving diffs between snapshots to a binary stream)<sup>[46]</sup>
- Incremental backup<sup>[47]</sup>
- Out-of-band data deduplication (requires userspace tools)<sup>[9]</sup>
- Ability to handle swap files and swap partitions

## Implemented but not recommended for production use

- Hierarchical per-subvolume quotas<sup>[48]</sup>
- RAID 5, RAID 6<sup>[49]</sup>

## Planned but not yet implemented

- In-band data deduplication<sup>[34]</sup>
- Online filesystem check<sup>[50]</sup>
- RAID with up to six parity devices, surpassing the reliability of RAID 5 and RAID 6<sup>[51]</sup>
- Object-level RAID 0, RAID 1, and RAID 10
- Encryption<sup>[8][52]</sup>
- Persistent read and write cache (L2ARC + ZIL, lvmcache, etc.)

In 2009, Btrfs was expected to offer a feature set comparable to ZFS, developed by Sun Microsystems.<sup>[53]</sup> After Oracle's acquisition of Sun in 2009, Mason and Oracle decided to continue with Btrfs development.<sup>[54]</sup>

## Cloning

Btrfs provides a *clone* operation that atomically creates a copy-on-write snapshot of a file. Such cloned files are sometimes referred to as reflinks, in light of the proposed associated Linux kernel system call.<sup>[55]</sup>

By cloning, the file system does not create a new link pointing to an existing inode; instead, it creates a new inode that initially shares the same disk blocks with the original file. As a result, cloning works only within the boundaries of the same Btrfs file system, but since version 3.6 of the Linux kernel it may cross the boundaries of subvolumes under certain circumstances.<sup>[56][57]</sup> The actual data blocks are not duplicated; at the same time, due to the copy-on-write (CoW) nature of Btrfs, modifications to any of the cloned files are not visible in the original file and vice versa.<sup>[58]</sup>

Cloning should not be confused with hard links, which are directory entries that associate multiple file names with actual files on a file system. While hard links can be taken as different names for the same file, cloning in Btrfs provides independent files that initially share all their disk blocks.<sup>[58][59]</sup>

Support for this Btrfs feature was added in version 7.5 of the GNU coreutils, via the `--reflink` option to the `cp` command.<sup>[60][61]</sup>

In addition to data cloning (FICLONE), Btrfs also supports for out-of-band deduplication via FIDEDUPERANGE. This functionality allows two files with (even partially) identical data to share storage.<sup>[62][9]</sup>

## Subvolumes and snapshots

A Btrfs subvolume can be thought of as a separate POSIX file namespace, mountable separately by passing `subvol` or `subvolid` options to the `mount(8)` ([https://man.cx/?page=mount\(8\)](https://man.cx/?page=mount(8))) utility. It can also be accessed by mounting the top-level subvolume, in which case subvolumes are visible and accessible as its subdirectories.<sup>[63]</sup>

Subvolumes can be created at any place within the file system hierarchy, and they can also be nested. Nested subvolumes appear as subdirectories within their parent subvolumes, similarly to the way a top-level subvolume presents its subvolumes as subdirectories. Deleting a subvolume is not possible until all subvolumes below it in the nesting hierarchy are deleted; as a result, top-level subvolumes cannot be deleted.<sup>[64]</sup>

Any Btrfs file system always has a default subvolume, which is initially set to be the top-level subvolume, and is mounted by default if no subvolume selection option is passed to `mount`. The default subvolume can be changed as required.<sup>[64]</sup>

A Btrfs snapshot is a subvolume that shares its data (and metadata) with some other subvolume, using Btrfs' copy-on-write capabilities, and modifications to a snapshot are not visible in the original subvolume. Once a writable snapshot is made, it can be treated as an alternate version of the original file system. For example, to roll back to a snapshot, a modified original subvolume needs to be unmounted and the snapshot needs to be mounted in its place. At that point, the original subvolume may also be deleted.<sup>[63]</sup>

The copy-on-write (CoW) nature of Btrfs means that snapshots are quickly created, while initially consuming very little disk space. Since a snapshot is a subvolume, creating nested snapshots is also possible. Taking snapshots of a subvolume is not a recursive process; thus, if a snapshot of a subvolume is created, every subvolume or snapshot that the subvolume already contains is mapped to an empty directory of the same name inside the snapshot.<sup>[63][64]</sup>

Taking snapshots of a directory is not possible, as only subvolumes can have snapshots. However, there is a workaround that involves reflinks spread across subvolumes: a new subvolume is created, containing cross-subvolume reflinks to the content of the targeted directory. Having that available, a snapshot of this new volume can be created.<sup>[56]</sup>

A subvolume in Btrfs is quite different from a traditional Logical Volume Manager (LVM) logical volume. With LVM, a logical volume is a separate block device, while a Btrfs subvolume is not and it cannot be treated or used that way.<sup>[63]</sup> Making `dd` or LVM snapshots of btrfs leads to dataloss if either the original or the copy is mounted while both are on the same computer.<sup>[65]</sup>

## Send–receive

Given any pair of subvolumes (or snapshots), Btrfs can generate a binary diff between them (by using the `btrfs send` command) that can be replayed later (by using `btrfs receive`), possibly on a different Btrfs file system. The send–receive feature effectively creates (and applies) a set of data modifications required for converting one subvolume into another.<sup>[46][66]</sup>

The send/receive feature can be used with regularly scheduled snapshots for implementing a simple form of file system replication, or for the purpose of performing incremental backups.<sup>[46][66]</sup>

## Quota groups

A *quota group* (or *qgroup*) imposes an upper limit to the space a subvolume or snapshot may consume. A new snapshot initially consumes no quota because its data is shared with its parent, but thereafter incurs a charge for new files and copy-on-write operations on existing files. When quotas are active, a quota group is automatically created with each new subvolume or snapshot. These initial quota groups are building blocks which can be grouped (with the `btrfs qgroup` command) into hierarchies to implement quota pools.<sup>[48]</sup>

Quota groups only apply to subvolumes and snapshots, while having quotas enforced on individual subdirectories, users, or user groups is not possible. However, workarounds are possible by using different subvolumes for all users or user groups that require a quota to be enforced.

## In-place conversion from ext2/3/4 and ReiserFS

As the result of having very little metadata anchored in fixed locations, Btrfs can warp to fit unusual spatial layouts of the backend storage devices. The `btrfs-convert` tool exploits this ability to do an in-place conversion of an ext2/3/4 or ReiserFS file system, by nesting the equivalent Btrfs metadata in its unallocated space—while preserving an unmodified copy of the original file system.<sup>[67]</sup>

The conversion involves creating a copy of the whole ext2/3/4 metadata, while the Btrfs files simply point to the same blocks used by the ext2/3/4 files. This makes the bulk of the blocks shared between the two filesystems before the conversion becomes permanent. Thanks to the copy-on-write nature of Btrfs, the original versions of the file data blocks are preserved during all file modifications. Until the conversion

becomes permanent, only the blocks that were marked as free in ext2/3/4 are used to hold new Btrfs modifications, meaning that the conversion can be undone at any time (although doing so will erase any changes made after the conversion to Btrfs).<sup>[67]</sup>

All converted files are available and writable in the default subvolume of the Btrfs. A sparse file holding all of the references to the original ext2/3/4 filesystem is created in a separate subvolume, which is mountable on its own as a read-only disk image, allowing both original and converted file systems to be accessed at the same time. Deleting this sparse file frees up the space and makes the conversion permanent.<sup>[67]</sup>

As of June 2015 and 4.x versions of the Linux kernel mainline, the in-place ext3/4 conversion was considered untested and rarely used.<sup>[67]</sup> The feature, however, was rewritten from scratch in 2016 for `btrfs-progs` 4.6.<sup>[44]</sup> and is considered stable since then.

In-place conversion from ReiserFS was introduced in September 2017 with kernel 4.13.<sup>[68]</sup>

## Union mounting / seed devices

When creating a new Btrfs, an existing Btrfs can be used as a read-only "seed" file system.<sup>[69]</sup> The new file system will then act as a copy-on-write overlay on the seed, as a form of union mounting. The seed can be later detached from the Btrfs, at which point the rebalancer will simply copy over any seed data still referenced by the new file system before detaching. Mason has suggested this may be useful for a Live CD installer, which might boot from a read-only Btrfs seed on an optical disc, rebalance itself to the target partition on the install disk in the background while the user continues to work, then eject the disc to complete the installation without rebooting.<sup>[70]</sup>

## Encryption

In his 2009 interview, Chris Mason stated that support for encryption was planned for Btrfs.<sup>[71]</sup> In the meantime, a workaround for combining encryption with Btrfs is to use a full-disk encryption mechanism such as dm-crypt / LUKS on the underlying devices and to create the Btrfs filesystem on top of that layer.

As of 2020, the developers were working to add keyed hash like HMAC (SHA256).<sup>[72]</sup>

## Checking and recovery

Unix systems traditionally rely on "fsck" programs to check and repair filesystems. This functionality is implemented via the `btrfs check` program. Since version 4.0 this functionality is deemed relatively stable. However, as of August 2017, the btrfs documentation suggests that it be used only after having tried other recovery methods.<sup>[73]</sup>

There is another tool, named `btrfs-restore`, that can be used to recover files from an unmountable filesystem, without modifying the broken filesystem itself (i.e., non-destructively).<sup>[74]</sup>

In normal use, Btrfs is mostly self-healing and can recover from broken root trees at mount time, thanks to making periodic data flushes to permanent storage, by default every 30 seconds. Thus, isolated errors will cause a maximum of 30 seconds of filesystem changes to be lost at the next mount.<sup>[75]</sup> This period can be changed by specifying a desired value (in seconds) for the `commit` mount option.<sup>[76][77]</sup>

## Design

---

Ohad Rodeh's original proposal at USENIX 2007 noted that B+ trees, which are widely used as on-disk data structures for databases, could not efficiently allow copy-on-write-based snapshots because its leaf nodes were linked together: if a leaf was copy-on-written, its siblings and parents would have to be as well, as would *their* siblings and parents and so on until the entire tree was copied. He suggested instead a modified B-tree (which has no leaf linkage), with a refcount associated to each tree node but stored in an ad hoc free map structure and certain relaxations to the tree's balancing algorithms to make them copy-on-write friendly. The result would be a data structure suitable for a high-performance object store that could perform copy-on-write snapshots, while maintaining good concurrency.<sup>[16]</sup>

At Oracle later that year, Chris Mason began work on a snapshot-capable file system that would use this data structure almost exclusively—not just for metadata and file data, but also recursively to track space allocation of the trees themselves. This allowed all traversal and modifications to be funneled through a single code path, against which features such as copy-on-write, checksumming and mirroring needed to be implemented only once to benefit the entire file system.<sup>[53]</sup>

Btrfs is structured as several layers of such trees, all using the same B-tree implementation. The trees store generic *items* sorted by a 136-bit key. The most significant 64 bits of the key are a unique *object id*. The middle eight bits are an item type field: its use is hardwired into code as an item filter in tree lookups. *Objects* can have multiple items of multiple types. The remaining (least significant) 64 bits are used in type-specific ways. Therefore, items for the same object end up adjacent to each other in the tree, grouped by type. By choosing certain key values, objects can further put items of the same type in a particular order.<sup>[53][3]</sup>

Interior tree nodes are simply flat lists of key-pointer pairs, where the pointer is the logical block number of a child node. Leaf nodes contain item keys packed into the front of the node and item data packed into the end, with the two growing toward each other as the leaf fills up.<sup>[53]</sup>

## File system tree

Within each directory, directory entries appear as *directory items*, whose least significant bits of key values are a CRC32C hash of their filename. Their data is a *location key*, or the key of the inode item it points to. Directory items together can thus act as an index for path-to-inode lookups, but are not used for iteration because they are sorted by their hash, effectively randomly permuting them. This means user applications iterating over and opening files in a large directory would thus generate many more disk seeks between non-adjacent files—a notable performance drain in other file systems with hash-ordered directories such as ReiserFS,<sup>[78]</sup> ext3 (with Htree-indexes enabled<sup>[79]</sup>) and ext4, all of which have TEA-hashed filenames. To avoid this, each directory entry has a *directory index item*, whose key value of the item is set to a per-directory counter that increments with each new directory entry. Iteration over these index items thus returns entries in roughly the same order as stored on disk.

Files with hard links in multiple directories have multiple reference items, one for each parent directory. Files with multiple hard links in the *same* directory pack all of the links' filenames into the same reference item. This was a design flaw that limited the number of same-directory hard links to however many could fit in a single tree block. (On the default block size of 4 KiB, an average filename length of 8 bytes and a per-filename header of 4 bytes, this would be less than 350.) Applications which made heavy use of multiple same-directory hard links, such as git, GNUS, Gnome and BackupPC were observed to fail at this limit.<sup>[80]</sup> The limit was eventually removed<sup>[81]</sup> (and as of October 2012 has been merged<sup>[82]</sup> pending release in Linux 3.7) by introducing spillover *extended reference items* to hold hard link filenames which do not otherwise fit.

## Extents

File data is kept outside the tree in *extents*, which are contiguous runs of disk data blocks. Extent blocks default to 4 KiB in size, do not have headers and contain only (possibly compressed) file data. In compressed extents, individual blocks are not compressed separately; rather, the compression stream spans the entire extent.

Files have *extent data items* to track the extents which hold their contents. The item's key value is the starting byte offset of the extent. This makes for efficient seeks in large files with many extents, because the correct extent for any given file offset can be computed with just one tree lookup.

Snapshots and cloned files share extents. When a small part of a large such extent is overwritten, the resulting copy-on-write may create three new extents: a small one containing the overwritten data, and two large ones with unmodified data on either side of the overwrite. To avoid having to re-write unmodified data, the copy-on-write may instead create *bookend extents*, or extents which are simply slices of existing extents. Extent data items allow for this by including an offset into the extent they are tracking: items for bookends are those with non-zero offsets.<sup>[3]</sup>

## Extent allocation tree

The *extent allocation tree* acts as an allocation map for the file system. Unlike other trees, items in this tree do not have object ids. They represent regions of space: their key values hold the starting offsets and lengths of the regions they represent.

The file system divides its allocated space into *block groups* which are variable-sized allocation regions that alternate between preferring metadata extents (tree nodes) and data extents (file contents). The default ratio of data to metadata block groups is 1:2. They are intended to use concepts of the Orlov block allocator to allocate related files together and resisting fragmentation by leaving free space between groups. (Ext3 block groups, however, have fixed locations computed from the size of the file system, whereas those in Btrfs are dynamic and created as needed.) Each block group is associated with a *block group item*. Inode items in the file system tree include a reference to their current block group.<sup>[3]</sup>

*Extent items* contain a back-reference to the tree node or file occupying that extent. There may be multiple back-references if the extent is shared between snapshots. If there are too many back-references to fit in the item, they spill out into individual *extent data reference items*. Tree nodes, in turn, have back-references to their containing trees. This makes it possible to find which extents or tree nodes are in any region of space by doing a B-tree range lookup on a pair of offsets bracketing that region, then following the back-references. For relocating data, this allows an efficient upwards traversal from the relocated blocks to quickly find and fix all downwards references to those blocks, without having to scan the entire file system. This, in turn, allows the file system to efficiently shrink, migrate, and defragment its storage online.

The extent allocation tree, as with all other trees in the file system, is copy-on-write. Writes to the file system may thus cause a cascade whereby changed tree nodes and file data result in new extents being allocated, causing the extent tree to itself change. To avoid creating a feedback loop, extent tree nodes which are still in memory but not yet committed to disk may be updated in-place to reflect new copy-on-written extents.

In theory, the extent allocation tree makes a conventional free-space bitmap unnecessary because the extent allocation tree acts as a B-tree version of a BSP tree. In practice, however, an in-memory red-black tree of page-sized bitmaps is used to speed up allocations. These bitmaps are persisted to disk (starting in Linux 2.6.37, via the `space_cache` mount option<sup>[83]</sup>) as special extents that are exempt from checksumming and copy-on-write.

## Checksum tree and scrubbing



CRC-32C checksums are computed for both data and metadata and stored as *checksum items* in a *checksum tree*. There is room for 256 bits of metadata checksums and up to a full node (roughly 4 KB or more) for data checksums. Btrfs has provisions for additional checksum algorithms to be added in future versions of the file system.<sup>[34][84]</sup>

There is one checksum item per contiguous run of allocated blocks, with per-block checksums packed end-to-end into the item data. If there are more checksums than can fit, they spill into another checksum item in a new leaf. If the file system detects a checksum mismatch while reading a block, it first tries to obtain (or create) a good copy of this block from another device – if internal mirroring or RAID techniques are in use.<sup>[85][86]</sup>

Btrfs can initiate an online check of the entire file system by triggering a file system scrub job that is performed in the background. The scrub job scans the entire file system for integrity and automatically attempts to report and repair any bad blocks it finds along the way.<sup>[85][87]</sup>

## Log tree

An fsync request commits modified data immediately to stable storage. fsync-heavy workloads (like a database or a virtual machine whose running OS *fsyncs* frequently) could potentially generate a great deal of redundant write I/O by forcing the file system to repeatedly copy-on-write and flush frequently modified parts of trees to storage. To avoid this, a temporary per-subvolume *log tree* is created to journal fsync-triggered copy-on-writes. Log trees are self-contained, tracking their own extents and keeping their own checksum items. Their items are replayed and deleted at the next full tree commit or (if there was a system crash) at the next remount.

## Chunk and device trees

Block devices are divided into *physical chunks* of 256 MB or more. Physical chunks across multiple devices can be mirrored or striped together into a single *logical chunk*. These logical chunks are combined into a single logical address space that the rest of the filesystem uses.

The *chunk tree* tracks this by storing each device therein as a *device item* and logical chunks as *chunk map items*, which provide a forward mapping from logical to physical addresses by storing their offsets in the least significant 64 bits of their key. Chunk map items can be one of several different types:

### single

1 logical to 1 physical chunk

### dup

1 logical chunk to 2 physical chunks on 1 block device

### raid0

N logical chunks to  $N \geq 2$  physical chunks across  $N \geq 2$  block devices

### raid1

1 logical chunk to 2 physical chunks across 2 out of  $N \geq 2$  block devices,<sup>[88]</sup> in contrast to conventional RAID 1 which has N physical chunks

### raid1c3

1 logical chunk to 3 physical chunks out of  $N \geq 3$  block devices

### raid1c4

1 logical chunk to 4 physical chunks out of  $N \geq 4$  block devices

### raid5

N (for  $N \geq 2$ ) logical chunks to N+1 physical chunks across N+1 block devices, with 1 physical chunk used as parity

### raid6

N (for  $N \geq 2$ ) logical chunks to N+2 physical chunks across N+2 block devices, with 2 physical chunks used as parity

*N* is the number of block devices still having free space when the chunk is allocated. If *N* is not large enough for the chosen mirroring/mapping, then the filesystem is effectively out of space.

## Relocation trees

Defragmentation, shrinking, and rebalancing operations require extents to be relocated. However, doing a simple copy-on-write of the relocating extent will break sharing between snapshots and consume disk space. To preserve sharing, an update-and-swap algorithm is used, with a special *relocation tree* serving as scratch space for affected metadata. The extent to be relocated is first copied to its destination. Then, by following backreferences upward through the affected subvolume's file system tree, metadata pointing to the old extent is progressively updated to point at the new one; any newly updated items are stored in the relocation tree. Once the update is complete, items in the relocation tree are swapped with their counterparts in the affected subvolume, and the relocation tree is discarded.<sup>[89]</sup>

## Superblock

All the file system's trees—including the chunk tree itself—are stored in chunks, creating a potential bootstrapping problem when mounting the file system. To bootstrap into a mount, a list of physical addresses of chunks belonging to the chunk and root trees are stored in the *superblock*.<sup>[90]</sup>

*Superblock mirrors* are kept at fixed locations:<sup>[91]</sup> 64 KiB into every block device, with additional copies at 64 MiB, 256 GiB and 1 PiB. When a superblock mirror is updated, its *generation number* is incremented. At mount time, the copy with the highest generation number is used. All superblock mirrors are updated in tandem, except in SSD mode which alternates updates among mirrors to provide some wear levelling.

## Commercial support

---

### Supported

- Fedora Workstation from version 33<sup>[92]</sup>
- Oracle Linux from version 7<sup>[93][94]</sup>
- SUSE Linux Enterprise Server from version 12<sup>[95][96]</sup>
- Synology DiskStation Manager (DSM) from version 6.0<sup>[97]</sup>
- ReactOS from version 0.4.10<sup>[98]</sup>

### No longer supported

- Btrfs was included as a "technology preview" in Red Hat Enterprise Linux 6 and 7;<sup>[22][31]</sup> it was removed in RHEL 8 in 2018.<sup>[32][99][100]</sup>

## See also

---

- APFS – a copy-on-write file system for macOS, iOS, tvOS, watchOS and audioOS
- Bcachefs
- Comparison of file systems

- HAMMER – DragonFly BSD's file system that uses B-trees, paired with checksums as a countermeasure for data corruption
- List of file systems
- ReFS – a copy-on-write file system for Windows Server 2012
- ZFS

## Notes

---

- a. This is the Btrfs' own on-disk size limit. The limit is reduced down to 8 EiB on 64-bit systems and 2 EiB on 32-bit systems due to Linux kernel's internal limits, unless kernel's CONFIG\_LBD configuration option (available since the 2.6.x kernel series) is enabled to remove these kernel limits.<sup>[101][102]</sup>
- b. Every item in Btrfs has a 64-bit identifier, which means the most files one can have on a Btrfs filesystem is  $2^{64}$ .

## References

---

1. "Btrfs Contributors at kernel.org" (<https://btrfs.wiki.kernel.org/index.php/Contributors>). kernel.org. 18 January 2016. Retrieved 20 January 2016.
2. "Suse Documentation: Storage Administration Guide – Large File Support in Linux" ([https://www.suse.com/documentation/sles11/stor\\_admin/data/sec\\_filesystems\\_lfs.html](https://www.suse.com/documentation/sles11/stor_admin/data/sec_filesystems_lfs.html)). SUSE. Retrieved 12 August 2015.
3. Mason, Chris. "Btrfs design" ([http://btrfs.wiki.kernel.org/articles/b/t/r/Btrfs\\_design.html](http://btrfs.wiki.kernel.org/articles/b/t/r/Btrfs_design.html)). *Btrfs wiki*. Retrieved 8 November 2011.
4. Jonathan Corbet (26 July 2010). "File creation times" (<https://lwn.net/Articles/397442/>). LWN.net. Retrieved 15 August 2015.
5. "On-disk Format - btrfs Wiki" ([https://btrfs.wiki.kernel.org/index.php/On-disk\\_Format#Basic\\_Structures](https://btrfs.wiki.kernel.org/index.php/On-disk_Format#Basic_Structures)). *btrfs.wiki.kernel.org*.
6. "btrfs Wiki" (<https://btrfs.wiki.kernel.org>). *kernel.org*. Retrieved 19 April 2015.
7. "Linux 4.14 - Linux Kernel Newbies" ([https://kernelnewbies.org/Linux\\_4.14](https://kernelnewbies.org/Linux_4.14)). *kernelnewbies.org*.
8. McPherson, Amanda (22 June 2009). "A Conversation with Chris Mason on BTRfs: the next generation file system for Linux" (<https://web.archive.org/web/20120627065427/http://www.linuxfoundation.org/news-media/blogs/browse/2009/06/conversation-chris-mason-btrfs-next-generation-file-system-linux>). Linux Foundation. Archived from the original (<http://www.linuxfoundation.org/news-media/blogs/browse/2009/06/conversation-chris-mason-btrfs-next-generation-file-system-linux>) on 27 June 2012. Retrieved 2009-09-01.
9. "Deduplication" (<https://btrfs.wiki.kernel.org/index.php/Deduplication>). *kernel.org*. Retrieved 19 April 2015.
10. "ReactOS 0.4.1 Released" (<https://reactos.org/project-news/reactos-041-released>). *reactos.org*. Retrieved 11 August 2016.
11. [http://streaming.oracle.com/ebn/podcasts/media/20209545\\_Oracle-Linux-7.mp4](http://streaming.oracle.com/ebn/podcasts/media/20209545_Oracle-Linux-7.mp4)
12. Henson, Valerie (31 January 2008). *Chunkfs: Fast file system check and repair* (<http://mirror.linux.org.au/pub/linux.conf.au/2008/Thu/mel8-262.ogg>). Melbourne, Australia. Event occurs at 18m 49s. Retrieved 5 February 2008. "It's called Butter FS or B-tree FS, but all the cool kids say Butter FS"
13. "Linux kernel commit changing stability status in fs/btrfs/Kconfig" (<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=4204617d142c0887e45fda2562cb5c58097b918e>). Retrieved 8 February 2019.

14. "22.2 About the Btrfs File System" ([https://web.archive.org/web/20180428201038/https://docs.oracle.com/cd/E52668\\_01/E54669/html/ol7-about-btrfs.html](https://web.archive.org/web/20180428201038/https://docs.oracle.com/cd/E52668_01/E54669/html/ol7-about-btrfs.html)). *Docs.Oracle.com*. Oracle. 2018. Archived from the original ([https://docs.oracle.com/cd/E52668\\_01/E54669/html/ol7-about-btrfs.html](https://docs.oracle.com/cd/E52668_01/E54669/html/ol7-about-btrfs.html)) on 28 April 2018. Retrieved 27 January 2021.
15. Kerner, Sean Michael (30 October 2008). "A Better File System for Linux?" (<http://www.internetnews.com/dev-news/article.php/3781676/A+Better+File+System+for+Linux.htm>). *InternetNews.com*. Archived (<https://web.archive.org/web/20110408185904/http://www.internetnews.com/dev-news/article.php/3781676/A%20Better%20File%20System%20for%20Linux.htm>) from the original on 8 April 2011. Retrieved 27 August 2020.
16. Rodeh, Ohad (2007). *B-trees, shadowing, and clones* (<https://www.usenix.org/legacy/events/lfs07/tech/rodeh.pdf>) (PDF). *USENIX Linux Storage & Filesystem Workshop*. Also Rodeh, Ohad (2008). "B-trees, shadowing, and clones". *ACM Transactions on Storage*. **3** (4): 1–27. doi:10.1145/1326542.1326544 (<https://doi.org/10.1145%2F1326542.1326544>).
17. "Lead Btrfs File-System Developers Join Facebook" ([https://www.phoronix.com/scan.php?page=news\\_item&px=MTUzNTE](https://www.phoronix.com/scan.php?page=news_item&px=MTUzNTE)). *phoronix.com*. Retrieved 19 April 2015.
18. Paul, Ryan (13 April 2009). "Panelists ponder the kernel at Linux Collaboration Summit" (<http://web.archive.org/web/20120617204105/http://arstechnica.com/information-technology/2009/04/linux-collaboration-summit-the-kernel-panel/>). *Ars Technica*. Archived from the original (<http://arstechnica.com/open-source/news/2009/04/linux-collaboration-summit-the-kernel-panel.ars>) on 17 June 2012. Retrieved 2009-08-22.
19. Ts'o, Theodore (1 August 2008). "Re: reiser4 for 2.6.27-rc1" (<https://lkml.org/lkml/2008/8/1/217>). *linux-kernel* (Mailing list). Retrieved 31 December 2010.
20. "Development timeline" ([https://web.archive.org/web/20081220083235/http://btrfs.wiki.kernel.org/index.php/Development\\_timeline](https://web.archive.org/web/20081220083235/http://btrfs.wiki.kernel.org/index.php/Development_timeline)). *Btrfs wiki*. 11 December 2008. Archived from the original ([http://btrfs.wiki.kernel.org/index.php/Development\\_timeline](http://btrfs.wiki.kernel.org/index.php/Development_timeline)) on 20 December 2008. Retrieved 5 November 2011.
21. Wuelfing, Britta (12 January 2009). "Kernel 2.6.29: Corbet Says Btrfs Next Generation Filesystem" (<http://www.linux-magazine.com/Online/News/Kernel-2.6.29-Corbet-Says-Btrfs-Next-Generation-Filesystem>). *Linux Magazine*. Retrieved 5 November 2011.
22. "Red Hat Enterprise Linux 6 documentation: Technology Previews" ([https://web.archive.org/web/20110528160211/http://docs.redhat.com/docs/en-US/Red\\_Hat\\_Enterprise\\_Linux/6/html/Technical\\_Notes/storage.html](https://web.archive.org/web/20110528160211/http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Technical_Notes/storage.html)). Archived from the original ([http://docs.redhat.com/docs/en-US/Red\\_Hat\\_Enterprise\\_Linux/6/html/Technical\\_Notes/storage.html#id4452791](http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Technical_Notes/storage.html#id4452791)) on 28 May 2011. Retrieved 21 January 2011.
23. "Fedora Weekly News Issue 276" ([https://fedoraproject.org/wiki/FWN/LatestIssue#What.27s\\_new\\_in\\_Fedora\\_15\\_.28Lovelock.29.3F](https://fedoraproject.org/wiki/FWN/LatestIssue#What.27s_new_in_Fedora_15_.28Lovelock.29.3F)). 25 May 2011.
24. "Debian 6.0 "Squeeze" released" (<http://www.debian.org/News/2011/20110205a.en.html>) (Press release). *Debian*. 6 February 2011. Retrieved 8 February 2011. "Support has also been added for the ext4 and Btrfs filesystems..."
25. "Linux kernel 3.0, Section 1.1. Btrfs: Automatic defragmentation, scrubbing, performance improvements" ([http://kernelnewbies.org/Linux\\_3.0#head-3e596e03408e1d32a7cc381d6f54e87feee22ee4](http://kernelnewbies.org/Linux_3.0#head-3e596e03408e1d32a7cc381d6f54e87feee22ee4)). *kernelnewbies.org*. 21 July 2011. Retrieved 5 April 2016.
26. Leemhuis, Thorsten (21 June 2011). "Kernel Log: Coming in 3.0 (Part 2) - Filesystems" (<http://www.h-online.com/open/features/Kernel-Log-Coming-in-3-0-Part-2-Filesystems-1263681.html>). *The H Open*. Retrieved 8 November 2011.
27. Varghese, Sam. "iTWire" (<http://www.itwire.com/business-it-news/open-source/62417-faebook-lures-top-btrfs-hackers>). *ITWire.com*. Retrieved 19 April 2015.
28. "Unbreakable Enterprise Kernel Release 2 has been released" (<https://blogs.oracle.com/linux/unbreakable-enterprise-kernel-release-2-has-been-released>). Retrieved 8 May 2019.
29. "SLES 11 SP2 Release Notes" ([http://www.novell.com/linux/releasenotes/x86\\_64/SUSE-SLES/11-SP2/#fate-306585](http://www.novell.com/linux/releasenotes/x86_64/SUSE-SLES/11-SP2/#fate-306585)). 21 August 2012. Retrieved 29 August 2012.

30. "SUSE Linux Enterprise Server 12 Release Notes" ([https://www.suse.com/releasenotes/x86\\_64/SUSE-SLES/12/#fate-317221](https://www.suse.com/releasenotes/x86_64/SUSE-SLES/12/#fate-317221)). 5 November 2015. Retrieved 20 January 2016.
31. "Red Hat Enterprise Linux 7.4 Release Notes, Chapter 53: Deprecated Functionality" ([https://web.archive.org/web/20170808013554/https://access.redhat.com/documentation/en-US/Red\\_Hat\\_Enterprise\\_Linux/7/html/7.4\\_Release\\_Notes/chap-Red\\_Hat\\_Enterprise\\_Linux-7.4\\_Release\\_Notes-Deprecated\\_Functionality.html](https://web.archive.org/web/20170808013554/https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/7.4_Release_Notes/chap-Red_Hat_Enterprise_Linux-7.4_Release_Notes-Deprecated_Functionality.html)). 1 August 2017. Archived from the original ([https://access.redhat.com/documentation/en-US/Red\\_Hat\\_Enterprise\\_Linux/7/html/7.4\\_Release\\_Notes/chap-Red\\_Hat\\_Enterprise\\_Linux-7.4\\_Release\\_Notes-Deprecated\\_Functionality.html](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/7.4_Release_Notes/chap-Red_Hat_Enterprise_Linux-7.4_Release_Notes-Deprecated_Functionality.html)) on 8 August 2017. Retrieved 15 August 2017.
32. "Considerations in adopting RHEL 8" ([https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/8/html/considerations\\_in\\_adopting\\_rhel\\_8/file-systems-and-storage\\_considerations-in-adopting-rhel-8#btrfs-has-been-removed\\_file-systems-and-storage](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/considerations_in_adopting_rhel_8/file-systems-and-storage_considerations-in-adopting-rhel-8#btrfs-has-been-removed_file-systems-and-storage)). *Product Documentation for Red Hat Enterprise Linux 8*. Red Hat. Retrieved 9 May 2019.
33. "Btrfs Coming to Fedora 33" (<https://fedoramagazine.org/btrfs-coming-to-fedora-33/>). *Fedora Magazine*. 24 August 2020. Retrieved 25 August 2020.
34. "Btrfs Wiki: Features" ([https://btrfs.wiki.kernel.org/index.php/Main\\_Page#Features](https://btrfs.wiki.kernel.org/index.php/Main_Page#Features)). *btrfs.wiki.kernel.org*. 27 November 2013. Retrieved 27 November 2013.
35. "Btrfs Wiki: Changelog" (<https://btrfs.wiki.kernel.org/index.php/Changelog>). *btrfs.wiki.kernel.org*. 29 May 2019. Retrieved 27 November 2013.
36. "Manpage btrfs-check" (<https://btrfs.wiki.kernel.org/index.php/Manpage/btrfs-check>).
37. "Using Btrfs with Multiple Devices" ([https://btrfs.wiki.kernel.org/index.php/Using\\_Btrfs\\_with\\_Multiple\\_Devices](https://btrfs.wiki.kernel.org/index.php/Using_Btrfs_with_Multiple_Devices)). *kernel.org*. 7 November 2013. Retrieved 20 November 2013.
38. "Compression" (<https://btrfs.wiki.kernel.org/index.php/Compression>). *kernel.org*. 25 June 2013. Retrieved 1 April 2014.
39. "Btrfs: add support for inode properties" (<https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=63541927c8d11d2686778b1e8ec71c14b4fd53e4>). *kernel.org*. 28 January 2014. Retrieved 1 April 2014.
40. "btrfs: Readonly snapshots" (<https://lwn.net/Articles/417617/>). Retrieved 12 December 2011.
41. "Save disk space on Linux by cloning files on Btrfs and OCFS2" (<https://blogs.oracle.com/otn/save-disk-space-on-linux-by-cloning-files-on-btrfs-and-ocfs2>). Retrieved 1 August 2017.
42. "Wiki FAQ: What checksum function does Btrfs use?" ([http://btrfs.wiki.kernel.org/index.php/FAQ#What\\_checksum\\_function\\_does\\_Btrfs\\_use.3F](http://btrfs.wiki.kernel.org/index.php/FAQ#What_checksum_function_does_Btrfs_use.3F)). *Btrfs wiki*. Retrieved 15 June 2009.
43. "Btrfs hilights in 5.5: new hashes" (<https://kdave.github.io/btrfs-hilights-5.5-new-hashes/>). Retrieved 29 August 2020.
44. "Btrfs progs release 4.6" (<https://www.spinics.net/lists/linux-btrfs/msg56040.html>). Retrieved 1 August 2017.
45. Mason, Chris (12 January 2009). "Btrfs changelog" ([https://web.archive.org/web/20120229050222/http://btrfs.ipv5.de/index.php?title=Changelog#Seed\\_Device\\_support](https://web.archive.org/web/20120229050222/http://btrfs.ipv5.de/index.php?title=Changelog#Seed_Device_support)). Archived from the original ([http://btrfs.ipv5.de/index.php?title=Changelog#Seed\\_Device\\_support](http://btrfs.ipv5.de/index.php?title=Changelog#Seed_Device_support)) on 29 February 2012. Retrieved 12 February 2012.
46. Corbet, Jonathan (11 July 2012), *Btrfs send/receive* (<https://lwn.net/Articles/506244/>), *LWN.net*, retrieved 14 November 2012
47. "Btrfs Wiki: Incremental Backup" ([https://btrfs.wiki.kernel.org/index.php/Incremental\\_Backup](https://btrfs.wiki.kernel.org/index.php/Incremental_Backup)). 27 May 2013. Retrieved 27 November 2013.
48. Jansen, Arne (2011), *Btrfs Subvolume Quota Groups* (<http://sensille.com/qgroups.pdf>) (PDF), *Strato AG*, retrieved 14 November 2012
49. btrfs (16 July 2016). "RAID 5/6" (<https://btrfs.wiki.kernel.org/index.php/RAID56>). *kernel.org*. Retrieved 1 October 2016.
50. Corbet, Jonathan (2 November 2011). "A btrfs update at LinuxCon Europe" (<https://lwn.net/Articles/465160/>). Retrieved 12 February 2012.


51. Mazzoleni, Andrea. "btrfs: lib: raid: New RAID library supporting up to six parities" (<https://lwn.net/Articles/579034/>). Retrieved 16 March 2014.
52. "Btrfs Project ideas" ([https://btrfs.wiki.kernel.org/index.php/Project\\_ideas](https://btrfs.wiki.kernel.org/index.php/Project_ideas)). 21 February 2013. Retrieved 21 February 2013.
53. Aurora, Valerie (22 July 2009). "A short history of btrfs" (<https://lwn.net/Articles/342892/>). LWN.net. Retrieved 5 November 2011.
54. Hilzinger, Marcel (22 April 2009). "Future of Btrfs Secured" (<http://www.linux-magazine.com/Online/News/Future-of-Btrfs-Secured>). *Linux Magazine*. Retrieved 5 November 2011.
55. Corbet, Jonathan (5 May 2009). "The two sides of reflink()" (<https://lwn.net/Articles/331808/>). LWN.net. Retrieved 17 October 2013.
56. "UseCases – btrfs documentation" (<https://btrfs.wiki.kernel.org/index.php/UseCases>). *kernel.org*. Retrieved 4 November 2013.
57. "btrfs: allow cross-subvolume file clone" (<https://github.com/torvalds/linux/commit/362a20c5e27614739c46707d1c5f55c214d164ce>). *github.com*. Retrieved 4 November 2013.
58. Lenz Grimmer (31 August 2011). "Save disk space on Linux by cloning files on Btrfs and OCFS2" ([https://blogs.oracle.com/OTNGarage/entry/save\\_disk\\_space\\_on\\_linux](https://blogs.oracle.com/OTNGarage/entry/save_disk_space_on_linux)). *oracle.com*. Retrieved 17 October 2013.
59. "Symlinks reference names, hardlinks reference meta-data and reflinks reference data" ([http://www.pixelbeat.org/docs/unix\\_links.html](http://www.pixelbeat.org/docs/unix_links.html)). *pixelbeat.org*. 27 October 2010. Retrieved 17 October 2013.
60. Meyering, Jim (20 August 2009). "GNU coreutils NEWS: Noteworthy changes in release 7.5" (<http://git.savannah.gnu.org/gitweb/?p=coreutils.git;a=blob;f=NEWS;h=601d73251c49ce36b39f9838aa818c740cf3a10a;hb=af1996dde2d0089117a9e5e7aa543c6e55474b77>). *savannah.gnu.org*. Retrieved 30 August 2009.
61. Scrivano, Giuseppe (1 August 2009). "cp: accept the --reflink option" (<http://git.savannah.gnu.org/gitweb/?p=coreutils.git;a=commit;h=a1d7469835371ded0ad8e3496bc5a5bebf94ccef>). *savannah.gnu.org*. Retrieved 2 November 2009.
62. `ioctl_fideduperange(2)` ([https://man7.org/linux/man-pages/man2/ioctl\\_fideduperange.2.html](https://man7.org/linux/man-pages/man2/ioctl_fideduperange.2.html)) – Linux Programmer's Manual – System Calls
63. "SysadminGuide – Btrfs documentation" (<https://btrfs.wiki.kernel.org/index.php/SysadminGuide>). *kernel.org*. Retrieved 31 October 2013.
64. "5.6 Creating Subvolumes and Snapshots [needs update]" ([http://docs.oracle.com/cd/E37670\\_01/E37355/html/ol\\_use\\_case3\\_btrfs.html](http://docs.oracle.com/cd/E37670_01/E37355/html/ol_use_case3_btrfs.html)). *oracle.com*. 2013. Retrieved 31 October 2013.
65. "Gotchas - btrfs Wiki" ([https://btrfs.wiki.kernel.org/index.php/Gotchas#Block-level\\_copies\\_of\\_devices](https://btrfs.wiki.kernel.org/index.php/Gotchas#Block-level_copies_of_devices)). *btrfs.wiki.kernel.org*.
66. "5.7 Using the Send/Receive Feature" ([http://docs.oracle.com/cd/E37670\\_01/E37355/html/ol\\_sendrecv\\_btrfs.html](http://docs.oracle.com/cd/E37670_01/E37355/html/ol_sendrecv_btrfs.html)). *oracle.com*. 2013. Retrieved 31 October 2013.
67. Mason, Chris (25 June 2015). "Conversion from Ext3 (Btrfs documentation)" ([https://btrfs.wiki.kernel.org/index.php/Conversion\\_from\\_Ext3](https://btrfs.wiki.kernel.org/index.php/Conversion_from_Ext3)). *kernel.org*. Retrieved 22 April 2016.
68. "btrfs-convert(8) manual page" (<https://btrfs.wiki.kernel.org/index.php/Manpage/btrfs-convert>). Retrieved 24 April 2018.
69. "Seed device" (<https://btrfs.wiki.kernel.org/index.php/Seed-device>).
70. Mason, Chris (5 April 2012), *Btrfs Filesystem: Status and New Features* (<http://linuxfoundation.ubicast.tv/videos/permalink/123/>), Linux Foundation, retrieved 16 November 2012

71. McPherson, Amanda (22 June 2009). "A Conversation with Chris Mason on BTRfs: the next generation file system for Linux" (<https://web.archive.org/web/20120627065427/http://www.linuxfoundation.org/news-media/blogs/browse/2009/06/conversation-chris-mason-btrfs-next-generation-file-system-linux>). Linux Foundation. Archived from the original (<http://www.linuxfoundation.org/news-media/blogs/browse/2009/06/conversation-chris-mason-btrfs-next-generation-file-system-linux>) on 27 June 2012. Retrieved 9 October 2014. "In future releases we plan to add online fsck, deduplication, encryption and other features that have been on admin wish lists for a long time."
72. Sterba, David. "authenticated file systems using HMAC(SHA256)" (<https://lore.kernel.org/linux-btrfs/3ca669b7-7447-5793-f231-32d5417bd8ee@suse.com/T/#m949c14afbe4485faf61bd6a568abfe21163bf5bd>). *Lore.Kernel.org*. Retrieved 25 April 2020.
73. "Btrfsck - btrfs Wiki" (<https://btrfs.wiki.kernel.org/index.php/Btrfsck>). *btrfs.wiki.kernel.org*.
74. "Restore - btrfs Wiki" (<https://btrfs.wiki.kernel.org/index.php/Restore>). *btrfs.wiki.kernel.org*.
75. "Problem FAQ - btrfs Wiki" ([https://btrfs.wiki.kernel.org/index.php/Problem\\_FAQ](https://btrfs.wiki.kernel.org/index.php/Problem_FAQ)). *kernel.org*. 31 July 2013. Retrieved 16 January 2014.
76. "kernel/git/torvalds/linux.git: Documentation: filesystems: add new btrfs mount options (Linux kernel source tree)" (<https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=906c176e541f89ed3c04d0e9af1c7cf7b3cc1adb>). *kernel.org*. 21 November 2013. Retrieved 6 February 2014.
77. "Mount options - btrfs Wiki" ([https://btrfs.wiki.kernel.org/index.php/Mount\\_options](https://btrfs.wiki.kernel.org/index.php/Mount_options)). *kernel.org*. 12 November 2013. Retrieved 16 January 2014.
78. Reiser, Hans (7 December 2001). "Re: Ext2 directory index: ALS paper and benchmarks" (<http://lkml.indiana.edu/hypermail/linux/kernel/0112.0/2019.html>). *ReiserFS developers mailing list*. Retrieved 28 August 2009.
79. Mason, Chris. "Acp" (<http://oss.oracle.com/~mason/acp/>). *Oracle personal web page*. Retrieved 5 November 2011.
80. "Hard Link Limitation" (<http://kerneltrap.org/mailarchive/linux-btrfs/2010/8/2/6885208/thread>). *kerneltrap.org*. 8 August 2010. Retrieved 14 November 2011.
81. Fasheh, Mark (9 October 2012). "btrfs: extended inode refs" (<https://archive.today/20130415062145/http://git.kernel.org/?p=linux/kernel/git/mason/linux-btrfs.git;a=commit;h=f186373fef005cee948a4a39e6a14c2e5f517298>). Archived from the original (<https://git.kernel.org/?p=linux/kernel/git/mason/linux-btrfs.git;a=commit;h=f186373fef005cee948a4a39e6a14c2e5f517298>) on 15 April 2013. Retrieved 7 November 2012.
82. Torvalds, Linus (10 October 2012). "Pull btrfs update from Chris Mason" (<https://archive.today/20130415043758/http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=72055425e53540d9d0e59a57ac8c9b8ce77b62d5>). *git.kernel.org*. Archived from the original (<http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=72055425e53540d9d0e59a57ac8c9b8ce77b62d5>) on 15 April 2013. Retrieved 7 November 2012.
83. Larabel, Michael (24 December 2010). "Benchmarks Of The Btrfs Space Cache Option" ([http://www.phoronix.com/scan.php?page=article&item=btrfs\\_space\\_cache&num=1](http://www.phoronix.com/scan.php?page=article&item=btrfs_space_cache&num=1)). *Phoronix*. Retrieved 16 November 2012.
84. "FAQ - btrfs Wiki: What checksum function does Btrfs use?" ([https://btrfs.wiki.kernel.org/index.php/FAQ#What\\_checksum\\_function\\_does\\_Btrfs\\_use.3F](https://btrfs.wiki.kernel.org/index.php/FAQ#What_checksum_function_does_Btrfs_use.3F)). The btrfs Project. Retrieved 22 November 2020.
85. Bierman, Margaret; Grimmer, Lenz (August 2012). "How I Use the Advanced Capabilities of Btrfs" (<http://www.oracle.com/technetwork/articles/servers-storage-admin/advanced-btrfs-1734952.html>). Retrieved 20 September 2013.
86. Salter, Jim (15 January 2014). "Bitrot and Atomic COWs: Inside "Next-Gen" Filesystems" (<http://arstechnica.com/information-technology/2014/01/bitrot-and-atomic-cows-inside-next-gen-file-systems/>). *Ars Technica*. Retrieved 15 January 2014.

87. Coekaerts, Wim (28 September 2011). "Btrfs Scrub – Go Fix Corruptions with Mirror Copies Please!" ([https://blogs.oracle.com/wim/entry/btrfs\\_scrub\\_go\\_fix\\_corruptions](https://blogs.oracle.com/wim/entry/btrfs_scrub_go_fix_corruptions)). Retrieved 20 September 2013.
88. "Manpage/MKFS.BTRFS - BTRFS Wiki" (<https://btrfs.wiki.kernel.org/index.php/Manpage/mkfs.btrfs#PROFILES>).
89. Mason, Chris; Rodeh, Ohad; Bacik, Josef (9 July 2012), *BTRFS: The Linux B-tree Filesystem* ([http://domino.watson.ibm.com/library/CyberDig.nsf/papers/6E1C5B6A1B6EDD9885257A38006B6130/\\$File/rj10501.pdf](http://domino.watson.ibm.com/library/CyberDig.nsf/papers/6E1C5B6A1B6EDD9885257A38006B6130/$File/rj10501.pdf)) (PDF), IBM Research, retrieved 12 November 2012
90. Mason, Chris (30 April 2008). "Multiple device support" ([https://web.archive.org/web/20110720220543/https://btrfs.wiki.kernel.org/index.php/Multiple\\_Device\\_Support](https://web.archive.org/web/20110720220543/https://btrfs.wiki.kernel.org/index.php/Multiple_Device_Support)). *Btrfs wiki*. Archived from the original ([http://btrfs.wiki.kernel.org/index.php/Multiple\\_Device\\_Support](http://btrfs.wiki.kernel.org/index.php/Multiple_Device_Support)) on 20 July 2011. Retrieved 5 November 2011.
91. Bartell, Sean (20 April 2010). "Re: Restoring BTRFS partition" (<http://kerneltrap.org/mailarchive/linux-btrfs/2010/4/20/6884623>). *linux-btrfs* (Mailing list).
92. "Fedora 33 is Officially Here!" (<https://fedoramagazine.org/announcing-fedora-33/>). 27 October 2020. Retrieved 28 October 2020.
93. "Oracle Now Supports Btrfs RAID5/6 On Their Unbreakable Enterprise Kernel - Phoronix" ([http://www.phoronix.com/scan.php?page=news\\_item&px=Btrfs-RAID56-UEK](http://www.phoronix.com/scan.php?page=news_item&px=Btrfs-RAID56-UEK)). *Phoronix.com*.
94. "Managing Btrfs in Oracle Linux 8" (<https://docs.oracle.com/en/operating-systems/oracle-linux/8/fsadmin/about-btrfs.html>). *docs.oracle.com*.
95. "SUSE Reaffirms Support for Btrfs" (<https://lwn.net/Articles/731848/>). *LWN.net*.
96. "SUSE Linux Enterprise Server 12 Release Notes" ([https://www.suse.com/releasenotes/x86\\_64/SUSE-SLES/12/](https://www.suse.com/releasenotes/x86_64/SUSE-SLES/12/)). *SUSE.com*. Retrieved 28 February 2021.
97. "Cloud Station White Paper" ([https://global.download.synology.com/download/Document/Software/WhitePaper/Package/CloudStation/All/enu/Synology\\_Cloud\\_Station\\_White\\_Paper-Based\\_on\\_DSM\\_6.0.pdf](https://global.download.synology.com/download/Document/Software/WhitePaper/Package/CloudStation/All/enu/Synology_Cloud_Station_White_Paper-Based_on_DSM_6.0.pdf)) (PDF). *Synology.com*. Synology. p. 11. Retrieved 2 April 2021. "Starting from DSM 6.0, data volumes can be formatted as Btrfs"
98. "File Systems" ([https://reactos.org/wiki/File\\_Systems](https://reactos.org/wiki/File_Systems)). *ReactOS.org*. Retrieved 28 February 2021.
99. "Btrfs has been deprecated in RHEL | Hacker News" (<https://news.ycombinator.com/item?id=14907771>). *News.YCombinator.com*.
00. "Red Hat Appears to Be Abandoning Their Btrfs Hopes - Phoronix" ([https://www.phoronix.com/scan.php?page=news\\_item&px=Red-Hat-Deprecates-Btrfs-Again](https://www.phoronix.com/scan.php?page=news_item&px=Red-Hat-Deprecates-Btrfs-Again)). *Phoronix.com*.
01. Andreas Jaeger (15 February 2005). "Large File Support in Linux" ([http://users.suse.com/~aj/linux\\_lfs.html](http://users.suse.com/~aj/linux_lfs.html)). *users.suse.com*. Retrieved 12 August 2015.
02. "Linux kernel configuration help for CONFIG\_LBD in 2.6.29 on x86" (<https://web.archive.org/web/20150906090823/http://kernel.xc.net/html/linux-2.6.29/x86/LBD>). *kernel.xc.net*. Archived from the original (<http://kernel.xc.net/html/linux-2.6.29/x86/LBD>) on 6 September 2015. Retrieved 12 August 2015.

## External links

---

- Official website (<https://btrfs.wiki.kernel.org/>) 
- I Can't Believe This is Butter! A tour of btrfs (<https://www.youtube.com/watch?v=hxWuaozpe2I>) on YouTube – a conference presentation by Avi Miller, an Oracle engineer
- Btrfs: Working with multiple devices (<https://lwn.net/Articles/577961/>) – LWN.net, December 2013, by Jonathan Corbet
- Marc's Linux Btrfs posts (<http://marc.merlins.org/perso/btrfs/>) – detailed insights into various Btrfs features



- [Btrfs overview \(http://marc.merlins.org/linux/talks/Btrfs-LC2014-JP/Btrfs.pdf\)](http://marc.merlins.org/linux/talks/Btrfs-LC2014-JP/Btrfs.pdf), LinuxCon 2014, by Marc Merlin
  - [File System Evangelist and Thought Leader: An Interview with Valerie Aurora \(http://www.linux-mag.com/id/7416/\)](http://www.linux-mag.com/id/7416/), *Linux Magazine*, 14 July 2009, by Jeffrey B. Layton
- 

Retrieved from "<https://en.wikipedia.org/w/index.php?title=Btrfs&oldid=1021962911>"

---

**This page was last edited on 7 May 2021, at 17:13 (UTC).**

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.