

NAME

btrfs-filesystem – command group that primarily does work on the whole filesystems

SYNOPSIS

btrfs filesystem <subcommand> <args>

DESCRIPTION

btrfs filesystem is used to perform several whole filesystem level tasks, including all the regular filesystem operations like resizing, space stats, label setting/getting, and defragmentation. There are other whole filesystem tasks like scrub or balance that are grouped in separate commands (`btrfs-scrub(8)` <>, `btrfs-balance(8)` <>).

SUBCOMMAND

df [options] <path>

Show a terse summary information about allocation of block group types of a given mount point. The original purpose of this command was a debugging helper. The output needs to be further interpreted and is not suitable for quick overview.

An example with description:

- device size: *1.9TiB*, one device, no RAID
- filesystem size: *1.9TiB*
- created with: **mkfs.btrfs -d single -m single**

```
$ btrfs filesystem df /path
Data, single: total=1.15TiB, used=1.13TiB
System, single: total=32.00MiB, used=144.00KiB
Metadata, single: total=12.00GiB, used=6.45GiB
GlobalReserve, single: total=512.00MiB, used=0.00B
```

- *Data*, *System* and *Metadata* are separate block group types. *GlobalReserve* is an artificial and internal emergency space, see below.
- *single* -- the allocation profile, defined at mkfs time
- *total* -- sum of space reserved for all allocation profiles of the given type, i.e. all *Data/single*. Note that it's not total size of filesystem.
- *used* -- sum of used space of the above, i.e. file extents, metadata blocks

GlobalReserve is an artificial and internal emergency space. It is used e.g. when the filesystem is full. Its *total* size is dynamic based on the filesystem size, usually not larger than 512MiB, *used* may fluctuate.

The *GlobalReserve* is a portion of *Metadata*. In case the filesystem metadata is exhausted, *GlobalReserve/total* + *Metadata/used* = *Metadata/total*. Otherwise there appears to be some unused space of *Metadata*.

Options

-b|--raw

raw numbers in bytes, without the *B* suffix

-h|--human-readable

print human friendly numbers, base 1024, this is the default

-H

print human friendly numbers, base 1000

--iec

select the 1024 base for the following options, according to the IEC standard

- si** select the 1000 base for the following options, according to the SI standard
- k|--kbytes**
show sizes in KiB, or kB with **--si**
- m|--mbytes**
show sizes in MiB, or MB with **--si**
- g|--gbytes**
show sizes in GiB, or GB with **--si**
- t|--tbytes**
show sizes in TiB, or TB with **--si**

If conflicting options are passed, the last one takes precedence.

defragment [options] <file>|<dir> [<file>|<dir>...]

Defragment file data on a mounted filesystem. Requires kernel 2.6.33 and newer.

If **-r** is passed, files in *dir* will be defragmented recursively (not descending to subvolumes, mount points and directory symlinks). The start position and the number of bytes to defragment can be specified by start and length using **-s** and **-l** options below. Extents bigger than value given by **-t** will be skipped, otherwise this value is used as a target extent size, but is only advisory and may not be reached if the free space is too fragmented. Use 0 to take the kernel default, which is 256KiB but may change in the future. You can also turn on compression in defragment operations.

Warning:

Defragmenting with Linux kernel versions < 3.9 or ≥ 3.14-rc2 as well as with Linux stable kernel versions ≥ 3.10.31, ≥ 3.12.12 or ≥ 3.13.4 will break up the reflinks of COW data (for example files copied with **cp --reflink**, snapshots or de-duplicated data). This may cause considerable increase of space usage depending on the broken up reflinks.

Note:

Directory arguments without **-r** do not defragment files recursively but will defragment certain internal trees (extent tree and the subvolume tree). This has been confusing and could be removed in the future.

For *start*, *len*, *size* it is possible to append units designator: *K*, *M*, *G*, *T*, *P*, or *E*, which represent KiB, MiB, GiB, TiB, PiB, or EiB, respectively (case does not matter).

Options

- c[<algo>]**
compress file contents while defragmenting. Optional argument selects the compression algorithm, *zlib* (default), *lzo* or *zstd*. Currently it's not possible to select no compression. See also section *EXAMPLES*.
- L|--level <level>**
Since kernel 6.15 the compression can also take the level parameter which will be used only for the defragmentation and overrides the eventual mount option compression level. Valid levels depend on the compression algorithms: *zlib* 1..9, *lzo* does not have any levels, *zstd* the standard levels 1..15 and also the realtime -1..-15.
- nocomp**
Do not compress while defragmenting, uncompress extents if needed.
- r**
defragment files recursively in given directories, does not descend to subvolumes or mount points

-f flush data for each file before going to the next file.

This will limit the amount of dirty data to current file, otherwise the amount accumulates from several files and will increase system load. This can also lead to ENOSPC if there's too much dirty data to write and it's not possible to make the reservations for the new data (i.e. how the COW design works).

-s <start>[kKmMgGtTpPeE]

defragmentation will start from the given offset, default is beginning of a file

-l <len>[kKmMgGtTpPeE]

defragment only up to *len* bytes, default is the file size

-t <size>[kKmMgGtTpPeE]

target extent size, do not touch extents bigger than *size*, default: 32MiB

The value is only advisory and the final size of the extents may differ, depending on the state of the free space and fragmentation or other internal logic. Reasonable values are from tens to hundreds of megabytes.

--step SIZE

Perform defragmentation in the range in *SIZE* steps and flush (**-f**) after each one. The range is default (the whole file) or given by **-s** and **-l**, split into the steps or done in one go if the step is larger. Minimum range size is 256KiB. With verbosity options the progress of defragmentation will be also printed.

-v (deprecated) alias for global **-v** option

du [options] <path> [<path>..]

Calculate disk usage of the target files using FIEMAP. For individual files, it will report a count of total bytes, and exclusive (not shared) bytes. We also calculate a 'set shared' value which is described below.

Each argument to **btrfs filesystem du** will have a *set shared* value calculated for it. We define each *set* as those files found by a recursive search of an argument (recursion descends to subvolumes but not mount points). The *set shared* value then is a sum of all shared space referenced by the set.

set shared takes into account overlapping shared extents, hence it isn't as simple as adding up shared extents.

Options

-s|--summarize

display only a total for each argument

--raw raw numbers in bytes, without the *B* suffix.

--human-readable

print human friendly numbers, base 1024, this is the default

--iec select the 1024 base for the following options, according to the IEC standard.

--si select the 1000 base for the following options, according to the SI standard.

--kbytes

show sizes in KiB, or kB with **--si**.

--mbytes

show sizes in MiB, or MB with **--si**.

--gbytes

show sizes in GiB, or GB with **--si**.

--tbytes

show sizes in TiB, or TB with **--si**.

label [<device>|<mountpoint>] [<newlabel>]

Show or update the label of a filesystem. This works on a mounted filesystem or a filesystem image.

The *newlabel* argument is optional. Current label is printed if the argument is omitted.

Note:

The maximum allowable length shall be less than 256 chars and must not contain a newline. The trailing newline is stripped automatically.

mkswapfile [-s size] file

Create a new file that's suitable and formatted as a swapfile. Default size is 2GiB, fixed page size 4KiB, minimum size is 40KiB.

A swapfile must be created in a specific way: NOCOW and preallocated. Subvolume containing a swapfile cannot be snapshotted and blocks of an activated swapfile cannot be balanced.

Swapfile creation can be achieved by standalone commands too. Activation needs to be done by command `swapon(8)` <<https://man7.org/linux/man-pages/man8/swapon.8.html>>. See also command **btrfs inspect-internal map-swapfile** and the Swapfile feature <> description.

Note:

The command is a simplified version of 'mkswap', if you want to set label, page size, or other parameters please use 'mkswap' proper.

Options**-s|--size SIZE**

Create swapfile of a given size SIZE (accepting k/m/g/e/p suffix).

-U|--uuid UUID

specify UUID to use, or a special value: clear (all zeros), random, time (time-based random)

resize [options] [<devid>:][+/-]<size>[kKmMgGtTpPeE][<devid>:]max <path>

Resize a mounted filesystem identified by *path*. A particular device can be resized by specifying a *devid*.

Note:

To resize a file containing a BTRFS image, please use the **--offline** flag.

The *devid* can be found in the output of **btrfs filesystem show** and defaults to 1 if not specified. The *size* parameter specifies the new size of the filesystem. If the prefix + or - is present the size is increased or decreased by the quantity *size*. If no units are specified, bytes are assumed for *size*. Optionally, the size parameter may be suffixed by one of the following unit designators: K, M, G, T, P, or E, which represent KiB, MiB, GiB, TiB, PiB, or EiB, respectively (case does not matter).

If *max* is passed, the filesystem will occupy all available space on the device respecting *devid* (remember, *devid* 1 by default).

The **resize** command does not manipulate the size of underlying partition. If you wish to enlarge/reduce a filesystem, you must make sure you expand the partition before enlarging the filesystem and shrink the partition after reducing the size of the filesystem. This can be done using `fdisk(8)` <<https://man7.org/linux/man-pages/man8/fdisk.8.html>> or `parted(8)` <<https://man7.org/>>

linux/man-pages/man8/parted.8.html> to delete the existing partition and recreate it with the new desired size. When recreating the partition make sure to use the same starting partition offset as before.

The size of the portion that the filesystem uses of an underlying device can be determined via the **btrfs filesystem show --raw** command on the filesystem's mount point (where it's given for each *dev*id after the string *size* or via the **btrfs inspect-internal dump-super** command on the specific device (where it's given as the value of *dev_item.total_bytes*, which is not to be confused with *total_bytes*). The value is also the address of the first byte not used by the filesystem.

Growing is usually instant as it only updates the size. However, shrinking could take a long time if there are data in the device area that's beyond the new end. Relocation of the data takes time.

Note that there's a lower limit on the new size (either specified as an absolute size or difference) that is checked by kernel and rejected eventually as invalid. Lower values will print a warning but still pass the request to kernel. The currently known value is 256MiB.

See also section *EXAMPLES*.

Options

--enqueue

wait if there's another exclusive operation running, otherwise continue

--offline

resize an unmounted (offline) filesystem

Warning:

The offline resize functionality currently supports **only increasing** the size of **single-device** filesystems. IOW, shrinking and multi-device filesystems are **not supported** with this option.

For filesystems stored in regular files, the file will be truncated to the new size as part of the resize operation. This flag is cannot be used together with **--enqueue** since offline resizing is synchronous.

show [options] [<path>|<uuid>|<device>|<label>]

Show the btrfs filesystem with some additional info about devices and space allocation.

If no option none of *path/uuid/device/label* is passed, information about all the BTRFS filesystems is shown, both mounted and unmounted.

Options

-m|--mounted

probe kernel for mounted BTRFS filesystems

-d|--all-devices

scan all devices under **/dev**, otherwise the devices list is extracted from the **/proc/partitions** file. This is a fallback option if there's no device node manager (like udev) available in the system.

--raw raw numbers in bytes, without the *B* suffix

--human-readable

print human friendly numbers, base 1024, this is the default

--iec select the 1024 base for the following options, according to the IEC standard

--si select the 1000 base for the following options, according to the SI standard

--kbytes
show sizes in KiB, or kB with **--si**

--mbytes
show sizes in MiB, or MB with **--si**

--gbytes
show sizes in GiB, or GB with **--si**

--tbytes
show sizes in TiB, or TB with **--si**

commit-stats

Print number of commits and time stats since mount. This is also available in `/sys/fs/btrfs/FSID/commit_stats`. The stats are not persistent and are collected since mount of the filesystem, one of them can be reset to zero (*Max commit duration*).

```

UUID: bd18ebe1-7e1d-414f-a3d5-2644388b51cc
Commit stats since mount:
  Total commits:                1133
  Last commit duration:         0ms
  Max commit duration:         10ms
  Total time spent in commit:   4543ms

```

Options

-z|--reset
print stats and reset 'max_commit_ms' (needs root)

sync <path>

Force a sync of the filesystem at *path*, similar to the `sync(1)` <<https://man7.org/linux/man-pages/man1/sync.1.html>> command. In addition, it starts cleaning of deleted subvolumes. To wait for the subvolume deletion to complete use the **btrfs subvolume sync** command.

usage [options] [<path> [<path>...]

Show detailed information about internal filesystem usage. This is supposed to replace the **btrfs filesystem df** command in the long run.

The level of detail can differ if the command is run under a regular or the root user (due to use of restricted ioctl). For both there's a summary section with information about space usage:

```

$ btrfs filesystem usage /path
WARNING: cannot read detailed chunk info, RAID5/6 numbers will be incorrect
Overall:
  Device size:                1.82TiB
  Device allocated:           1.17TiB
  Device unallocated:         669.99GiB
  Device missing:              0.00B
  Device slack:                1.00GiB
  Used:                        1.14TiB
  Free (estimated):           692.57GiB    (min: 692.57GiB)
  Free (statfs, df)           692.57GiB
  Data ratio:                  1.00
  Metadata ratio:              1.00
  Global reserve:              512.00MiB    (used: 0.00B)
  Multiple profiles:           no

```

- *Device size* -- sum of raw device capacity available to the filesystem, note that this may not be the same as the total device size (the difference is accounted as slack)
- *Device allocated* -- sum of total space allocated for data/metadata/system profiles, this also accounts space reserved but not yet used for extents
- *Device unallocated* -- the remaining unallocated space for future allocations (difference of the above two numbers)
- *Device missing* -- sum of capacity of all missing devices
- *Device slack* -- sum of slack space on all devices (difference between entire device size and the space occupied by filesystem)
- *Used* -- sum of the used space of data/metadata/system profiles, not including the reserved space
- *Free (estimated)* -- approximate size of the remaining free space usable for data, including currently allocated space and estimating the usage of the unallocated space based on the block group profiles, the *min* is the lower bound of the estimate in case multiple profiles are present
- *Free (statfs, df)* -- the amount of space available for data as reported by the **statfs/statvfs** syscall, also returned as *Avail* in the output of **df**. The value is calculated in a different way and may not match the estimate in some cases (e.g. multiple profiles).
- *Data ratio* -- ratio of total space for data including redundancy or parity to the effectively usable data space, e.g. single is 1.0, RAID1 is 2.0 and for RAID5/6 it depends on the number of devices
- *Metadata ratio* -- ditto, for metadata
- *Global reserve* -- portion of metadata currently used for global block reserve, used for emergency purposes (like deletion on a full filesystem)
- *Multiple profiles* -- what block group types (data, metadata) have more than one profile (single, raid1, ...), see `btrfs(5)` <> section **FILESYSTEMS WITH MULTIPLE PROFILES** <# `man-btrfs5-filesystem-with-multiple-profiles`>.

And on a zoned filesystem there are two more lines in the *Device* section:

```
Device zone unusable:      5.13GiB
Device zone size:         256.00MiB
```

- *Device zone unusable* -- sum of of space that's been used in the past but now is not due to COW and not referenced anymore, the chunks have to be reclaimed and zones reset to make it usable again
- *Device zone size* -- the reported zone size of the host-managed device, same for all devices

The root user will also see stats broken down by block group types:

```
Data, single: Size:1.15TiB, Used:1.13TiB (98.26%)
/dev/sdb      1.15TiB

Metadata, single: Size:12.00GiB, Used:6.45GiB (53.75%)
/dev/sdb      12.00GiB

System, single: Size:32.00MiB, Used:144.00KiB (0.44%)
/dev/sdb      32.00MiB

Unallocated:
/dev/sdb      669.99GiB
```

Data is block group type, *single* is block group profile, *Size* is total size occupied by this type, *Used* is the actually used space, the percent is ratio of *Used/Size*. The *Unallocated* is remaining space.

Options

- b|--raw**
raw numbers in bytes, without the *B* suffix
- h|--human-readable**
print human friendly numbers, base 1024, this is the default
- H** print human friendly numbers, base 1000
- iec** select the 1024 base for the following options, according to the IEC standard
- si** select the 1000 base for the following options, according to the SI standard
- k|--kbytes**
show sizes in KiB, or kB with **--si**
- m|--mbytes**
show sizes in MiB, or MB with **--si**
- g|--gbytes**
show sizes in GiB, or GB with **--si**
- t|--tbytes**
show sizes in TiB, or TB with **--si**
- T** show data in tabular format

If conflicting options are passed, the last one takes precedence.

EXAMPLES

```
$ btrfs filesystem defrag -v -r dir/
```

Recursively defragment files under **dir/**, print files as they are processed. The file names will be printed in batches, similarly the amount of data triggered by defragmentation will be proportional to last N printed files. The system dirty memory throttling will slow down the defragmentation but there can still be a lot of IO load and the system may stall for a moment.

```
$ btrfs filesystem defrag -v -r -f dir/
```

Recursively defragment files under **dir/**, be verbose and wait until all blocks are flushed before processing next file. You can note slower progress of the output and lower IO load (proportional to currently defragmented file).

```
$ btrfs filesystem defrag -v -r -f -clzo dir/
```

Recursively defragment files under **dir/**, be verbose, wait until all blocks are flushed and force file compression.

```
$ btrfs filesystem defrag -v -r -t 64M dir/
```

Recursively defragment files under **dir/**, be verbose and try to merge extents to be about 64MiB. As stated above, the success rate depends on actual free space fragmentation and the final result is not guaranteed to meet the target even if run repeatedly.

```
$ btrfs filesystem resize -1G /path
```


\$ btrfs filesystem resize 1:-1G /path

Shrink size of the filesystem's device id 1 by 1GiB. The first syntax expects a device with id 1 to exist, otherwise fails. The second is equivalent and more explicit. For a single-device filesystem it's typically not necessary to specify the devid though.

\$ btrfs filesystem resize max /path**\$ btrfs filesystem resize 1:max /path**

Let's assume that devid 1 exists and the filesystem does not occupy the whole block device, e.g. it has been enlarged and we want to grow the filesystem. By simply using *max* as size we will achieve that.

Note:

There are two ways to minimize the filesystem on a given device. The **btrfs inspect-internal min-dev-size** command, or iteratively shrink in steps.

EXIT STATUS

btrfs filesystem returns a zero exit status if it succeeds. Non zero is returned in case of failure.

AVAILABILITY

btrfs is part of **btrfs-progs**. Please refer to the documentation at <<https://btrfs.readthedocs.io>>.

SEE ALSO

btrfs-subvolume(8) <>, **mkfs.btrfs(8)** <>