BTRFS
stable

Search docs

OVERVIEW
Introduction
Status
Manual pages
btrfs(8)
btrfs(5)
btrfs-balance(8)
btrfs-check(8)
btrfs-convert(8)
btrfs-device(8)
btrfs-filesystem(8)
btrfs-find-root(8)
btrfs-image(8)
btrfs-inspect-internal(8)
btrfs-ioctl(2)
btrfs-map-logical(8)
btrfs-property(8)
btrfs-qgroup(8)
btrfs-quota(8)
btrfs-receive(8)
btrfs-replace(8)
btrfs-rescue(8)
btrfs-restore(8)
btrfs-scrub(8)
SYNOPSIS
DESCRIPTION
SUBCOMMAND
EXIT STATUS
AVAILABILITY
SEE ALSO
btrfs-select-super(8)
btrfs-send(8)
btrfs-subvolume(8)
btrfstune(8)
fsck.btrfs(8)
mkfs.btrfs(8)
Administration
Hardware considerations
Changes (feature/version)
Changes (kernel/version)
Changes (btrfs-progs)
Contributors
Glossary
Installation instructions
Source repositories
Interoperability

FEATURES
Common Linux features
Custom ioctls
Auto-repair on read
Balance
Compression
Checksumming
Convert
Deduplication
Defragmentation
Inline files
Quota groups
Reflink
Resize
Scrub
Seeding device
Send/receive
Subpage support
Subvolumes
Swapfile
Tree checker
Trim/discard
Volume management
Zoned mode

DEVELOPER DOCUMENTATION
Development notes
Developer's FAQ
Conventions and style for documentation
Experimental features
Btrfs design
Btrees
On-disk Format
Send stream format
JSON output
Internal APIs
Release checklist
Pull request review workflow
Command line, formatting, UI guidelines
btrfs-ioctl(2)

TODO

btrfs-scrub(8)

SYNOPSIS

btrfs scrub <subcommand> <args>

DESCRIPTION

Scrub is a validation pass over all filesystem data and metadata that detects data checksum errors, basic super block errors, basic metadata block header errors, and disk read errors.

Scrub is done on a per-device base, if a device is specified to **btrfs scrub start**, then only that device will be scrubbed. Although btrfs will also try to read other device to find a good copy, if the mirror on that specified device failed to be read or pass verification.

If a path of btrfs is specified to **btrfs scrub start**, btrfs will scrub all devices in parallel.

On filesystems that use replicated block group profiles (e.g. RAID1), read-write scrub will also automatically repair any damage by copying verified good data from one of the other replicas.

Such automatic repair is also carried out when reading metadata or data from a read-write mounted filesystem.

> Warning
>
> As currently implemented, setting the `NOCOW` file attribute (by **chattr +C**) on a file implicitly enables `NODATASUM`. This means that while metadata for these files continues to be validated and corrected by scrub, the actual file data is not.
>
> Furthermore, btrfs does not currently mark missing or failed disks as unreliable, so will continue to load-balance reads to potentially damaged replicas. This is not a problem normally because damage is detected by checksum validation, but because `NOCOW` files are not protected by checksums, btrfs has no idea which mirror is good thus it can return the bad contents to the user space tool.
>
> Detecting and recovering from such failure requires manual intervention.
>
> Notably, systemd sets +C on journals by default, and libvirt ≥ 6.6 sets +C on storage pool directories by default. Other applications or distributions may also set `+C` to try to improve performance.

> Note
>
> Scrub is not a filesystem checker (fsck, btrfs-check(8)). It can only detect filesystem damage using the checksum validation, and it can only repair filesystem damage by copying from other known good replicas.
>
> btrfs-check(8) performs more exhaustive checking and can sometimes be used, with expert guidance, to rebuild certain corrupted filesystem structures in the absence of any good replica.

> Note
>
> Read-only scrub on a read-write filesystem will cause some writes into the filesystem.
>
> This is due to the design limitation to prevent race between marking block group read-only and writing back block group items.
>
> To avoid any writes from scrub, one has to run read-only scrub on read-only filesystem.

The user is supposed to run it manually or via a periodic system service. The recommended period is a month but it could be less. The estimated device bandwidth utilization is about 80% on an idle filesystem.

The scrubbing status is recorded in `/var/lib/btrfs/` in textual files named *scrub.status.UUID* for a filesystem identified by the given UUID. (Progress state is communicated through a named pipe in file *scrub.progress.UUID* in the same directory.) The status file is updated every 5 seconds. A resumed scrub will continue from the last saved position.

Scrub can be started only on a mounted filesystem, though it's possible to scrub only a selected device. See btrfs scrub start for more.

Bandwidth and IO limiting

> Note
>
> The ionice(1) may not be generally supported by all IO schedulers and the options to **btrfs scrub start** may not work as expected.

In the past when the CFQ IO scheduler was generally used the ionice(1) syscalls set the priority to *idle* so the IO would not interfere with regular IO. Since the kernel 5.0 the CFQ is not available.

The IO scheduler known to support that is BFQ, but first read the documentation before using it!

For other commonly used schedulers like mq-deadline it's recommended to use *cgroup2 IO controller* which could be managed by e.g. *systemd* (documented in `systemd.resource-control`). However, starting scrub like that is not yet completely straightforward. The IO controller must know the physical device of the filesystem and create a slice so all processes started from that belong to the same accounting group.

```
$ systemd-run -p "IOReadBandwidthMax=/dev/sdx 10M" btrfs scrub start -B /
```

Since linux 5.14 it's possible to set the per-device bandwidth limits in a BTRFS-specific way using files `/sys/fs/btrfs/FSID/devinfo/DEVID/scrub_speed_max`. This setting is not persistent, lasts until the filesystem is unmounted. Currently set limits can be displayed by command btrfs scrub limit.

```
$ echo 100m > /sys/fs/btrfs/9b5fd16e-1b64-4f9b-904a-74e74c0bbadc/devinfo/1/scrub_speed_max
$ btrfs scrub limit /
UUID: 9b5fd16e-1b64-4f9b-904a-74e74c0bbadc
Id      Limit      Path
--  ---------  --------
 1  100.00MiB  /dev/sdx
```

SUBCOMMAND

cancel <path>|<device>

If a scrub is running on the filesystem identified by *path* or *device*, cancel it.

If a *device* is specified, the corresponding filesystem is found and **btrfs scrub cancel** behaves as if it was called on that filesystem. The progress is saved in the status file so **btrfs scrub resume** can continue from the last position.

limit [options] <path>

Show or set scrub limits on devices of the given filesystem.

`Options`

-d|--devid DEVID

select the device by DEVID to apply the limit

-l|--limit SIZE

set the limit of the device to SIZE (size units with suffix), or 0 to reset to *unlimited*

-a|--all

apply the limit to all devices

stable

BTRFS
stable
Search docs
OVERVIEW
Introduction
Status
Manual pages
btrfs(8)
btrfs(5)
btrfs-balance(8)
btrfs-check(8)
btrfs-convert(8)
btrfs-device(8)
btrfs-filesystem(8)
btrfs-find-root(8)
btrfs-image(8)
btrfs-inspect-internal(8)
btrfs-ioctl(2)
btrfs-map-logical(8)
btrfs-property(8)
btrfs-qgroup(8)
btrfs-quota(8)
btrfs-receive(8)
btrfs-replace(8)
btrfs-rescue(8)
btrfs-restore(8)
btrfs-scrub(8)
SYNOPSIS
DESCRIPTION
SUBCOMMAND
EXIT STATUS
AVAILABILITY
SEE ALSO
btrfs-select-super(8)
btrfs-send(8)
btrfs-subvolume(8)
btrfstune(8)
fsck.btrfs(8)
mkfs.btrfs(8)
Administration
Hardware considerations
Changes (feature/version)
Changes (kernel/version)
Changes (btrfs-progs)
Contributors
Glossary
Installation instructions
Source repositories
Interoperability
FEATURES
Common Linux features
Custom ioctls
Auto-repair on read
Balance
Compression
Checksumming
Convert
Deduplication
Defragmentation
Inline files
Quota groups
Reflink
Resize
Scrub
Seeding device
Send/receive
Subpage support
Subvolumes
Swapfile
Tree checker
Trim/discard
Volume management
Zoned mode
DEVELOPER DOCUMENTATION
Development notes
Developer's FAQ
Conventions and style for documentation
Experimental features
Btrfs design
Btrees
On-disk Format
Send stream format
JSON output
Internal APIs
Release checklist
Pull request review workflow
Command line, formatting, UI guidelines
btrfs-ioctl(2)
TODO

`--raw`

print all numbers raw values 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

resume [-BdqrR] <path>|<device>

Resume a cancelled or interrupted scrub on the filesystem identified by *path* or on a given *device*. The starting point is read from the status file if it exists.

This does not start a new scrub if the last scrub finished successfully.

`Options`

see **scrub start**.

start [options] <path>|<device>

Start a scrub on all devices of the mounted filesystem identified by *path* or on a single *device*. If a scrub is already running, the new one will not start. A device of an unmounted filesystem cannot be scrubbed this way.

Without options, scrub is started as a background process. The automatic repairs of damaged copies are performed by default for block group profiles with redundancy. No-repair can be enabled by option *-r*.

`Options`

`-B`

do not background and print scrub statistics when finished

`-d`

print separate statistics for each device of the filesystem (*-B* only) at the end

`-r`

run in read-only mode, do not attempt to correct anything, can be run on a read-only filesystem

Note that a read-only scrub on a read-write filesystem can still cause writes into the filesystem due to some internal limitations. Only a read-only scrub on a read-only filesystem can avoid writes from scrub.

`-R`

raw print mode, print full data instead of summary

`--limit <limit>`

set the scrub throughput limit for each device.

If the scrub is for the whole fs, it's the same as **btrfs scrub limit -a -l <value>**. If the scrub is for a single device, it's the same as **btrfs scrub limit -d <devid> -l <value>**.

The value is bytes per second, and accepts the usual KMGT prefixes. After the scrub is finished, the throughput limit will be reset to the old value of each device.

`-f`

force starting new scrub even if a scrub is already running, this can useful when scrub status file is damaged and reports a running scrub although it is not, but should not normally be necessary

`Deprecated options`

`-c <ioprio_class>`

set IO priority class (see ionice(1) manual page) if the IO scheduler configured for the device supports ionice. This is only supported by BFQ or Kyber but is *not* supported by mq-deadline. Please read the section about IO limiting.

`-n <ioprio_classdata>`

set IO priority classdata (see ionice(1) manpage)

`-q`

(deprecated) alias for global *-q* option

status [options] <path>|<device>

Show status of a running scrub for the filesystem identified by *path* or for the specified *device*.

If no scrub is running, show statistics of the last finished or cancelled scrub for that filesystem or device.

`Options`

`-d`

print separate statistics for each device of the filesystem

`-R`

print all raw statistics without postprocessing as returned by the status ioctl

`--raw`

print all numbers raw values 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`

stable

BTRFS
stable

Search docs

OVERVIEW
Introduction
Status
Manual pages
btrfs(8)
btrfs(5)
btrfs-balance(8)
btrfs-check(8)
btrfs-convert(8)
btrfs-device(8)
btrfs-filesystem(8)
btrfs-find-root(8)
btrfs-image(8)
btrfs-inspect-internal(8)
btrfs-ioctl(2)
btrfs-map-logical(8)
btrfs-property(8)
btrfs-qgroup(8)
btrfs-quota(8)
btrfs-receive(8)
btrfs-replace(8)
btrfs-rescue(8)
btrfs-restore(8)
btrfs-scrub(8)
SYNOPSIS
DESCRIPTION
SUBCOMMAND
EXIT STATUS
AVAILABILITY
SEE ALSO
btrfs-select-super(8)
btrfs-send(8)
btrfs-subvolume(8)
btrfstune(8)
fsck.btrfs(8)
mkfs.btrfs(8)
Administration
Hardware considerations
Changes (feature/version)
Changes (kernel/version)
Changes (btrfs-progs)
Contributors
Glossary
Installation instructions
Source repositories
Interoperability
FEATURES
Common Linux features
Custom ioctls
Auto-repair on read
Balance
Compression
Checksumming
Convert
Deduplication
Defragmentation
Inline files
Quota groups
Reflink
Resize
Scrub
Seeding device
Send/receive
Subpage support
Subvolumes
Swapfile
Tree checker
Trim/discard
Volume management
Zoned mode
DEVELOPER DOCUMENTATION
Development notes
Developer's FAQ
Conventions and style for documentation
Experimental features
Btrfs design
Btrees
On-disk Format
Send stream format
JSON output
Internal APIs
Release checklist
Pull request review workflow
Command line, formatting, UI guidelines
btrfs-ioctl(2)
TODO

show sizes in TiB, or TB with --si

A status on a filesystem without any error looks like the following:

```
# btrfs scrub start /
# btrfs scrub status /
UUID:             76fac721-2294-4f89-a1af-620cde7a1980
Scrub started:    Wed Apr 10 12:34:56 2023
Status:           running
Duration:         0:00:05
Time left:        0:00:05
ETA:              Wed Apr 10 12:35:01 2023
Total to scrub:   28.32GiB
Bytes scrubbed:   13.76GiB  (48.59%)
Rate:             2.75GiB/s
Error summary:    no errors found
```

With some errors found:

```
Error summary:    csum=72
  Corrected:      2
  Uncorrectable:  72
  Unverified:     0
```

- *Corrected* -- number of bad blocks that were repaired from another copy
- *Uncorrectable* -- errors detected at read time but not possible to repair from other copy
- *Unverified* -- transient errors, first read failed but a retry succeeded, may be affected by lower layers that group or split IO requests
- *Error summary* -- followed by a more detailed list of errors found
  - *csum* -- checksum mismatch
  - *super* -- super block errors, unless the error is fixed immediately, the next commit will overwrite superblock
  - *verify* -- metadata block header errors
  - *read* -- blocks can't be read due to IO errors

It's possible to set a per-device limit via file `sysfs/fs/btrfs/FSID/devinfo/scrub_speed_max`. In that case the limit is printed on the *Rate:* line if option *-d* is specified, or without it on a single-device filesystem. Read more about tat in section about scrub IO limiting.

```
Rate:             989.0MiB/s (limit 1.0G/s)
```

On a multi-device filesystem with at least one device limit the overall stats cannot print the limit without *-d* so there's a not that some limits are set:

```
Rate:             36.37MiB/s (some device limits set)
```

EXIT STATUS

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

1

scrub couldn't be performed

2

there is nothing to resume

3

scrub found uncorrectable errors

AVAILABILITY

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

SEE ALSO

mkfs.btrfs(8)

Previous Next

stable