NAME

btrfs-rescue − recover a damaged btrfs filesystem

SYNOPSIS

btrfs rescue <subcommand> <args>

DESCRIPTION

A set of commands that are targeting to fix a specific problem and may not suitable for :doc`btrfs−check`.

Note:

All the subcommands cannot handle an ongoing replace/balance, please finish/cancel the replace/balance before using this command.

SUBCOMMAND

chunk−recover [options] <device>

Recover the chunk tree by scanning the devices

Options

−y assume an answer of *yes* to all questions.

−h help.

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

Note:

Since **chunk−recover** will scan the whole device, it will be very slow especially if executed on a large device.

fix−device−size <device>

Fix device size and super block total bytes values that do not match.

Kernel 4.11 starts to check the device size more strictly and this might mismatch the stored value of total bytes. See the exact error message below. Newer kernel will refuse to mount the filesystem where the values do not match. This error is not fatal and can be fixed. This command will fix the device size values if possible.

```
BTRFS error (device sdb): super_total_bytes 92017859088384 mismatch with fs
```

The mismatch may also exhibit as a kernel warning:

```
WARNING: CPU: 3 PID: 439 at fs/btrfs/ctree.h:1559 btrfs_update_device+0x1c5
```

fix−data−checksum <device>

Selectively fix data checksum mismatch.

There is a long existing problem that if a user space program is doing direct IO and modifies the buffer before the write back finished, it can lead to data checksum mismatches.

This problem is known but not fixed until upstream release v6.15 (backported to older kernels). So it's possible to hit false data checksum mismatch for any long running btrfs.

In that case this program can be utilized to repair such problem.

Options

−r|−−readonly

readonly mode, only scan for and report data checksum mismatches, do not repair

−i|−−interactive
interactive mode, ask for how to repair, ignore the errors by default

−m|−−mirror <num>
use specified mirror to update the checksum item for all corrupted blocks.

The value must be >= 1, and if the corrupted block has fewer mirrors than the value, the mirror number will be **num % (num_mirrors + 1)**.

clear−ino−cache <device>
Remove leftover items pertaining to the deprecated *inode number cache* feature.

The feature enabled by mount option *inode_cache* has been completely removed in 5.11 kernel.

clear−space−cache <v1|v2> <device>
Completely remove the on−disk data of free space cache of given version.

Especially for v1 free space cache, *clear_cache* mount option would only remove the cache for updated block groups, the remaining would not be removed. Thus this command is provided to manually clear the free space cache.

clear−uuid−tree <device>
Clear the UUID tree, so that kernel can regenerate it at next read−write mount.

Since kernel v4.16 there are more sanity check performed, and sometimes non−critical trees like UUID tree can cause problems and reject the mount. In such case, clearing UUID tree may make the filesystem to be mountable again without much risk as it's built from other trees. See also mount option rescan_uuid_tree (in btrfs−man5) <btrfs-man5//#mount-option-rescan-uuid-tree>.

super−recover [options] <device>
Recover bad superblocks from good copies.

Options

−y assume an answer of *yes* to all questions.

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

zero−log <device>
Clear the filesystem log tree.

This command will clear the filesystem log tree. This may fix a specific set of problem when the filesystem mount fails during log replay. See below for sample stack traces that may show up in system log.

Note:
Clearing the log may lead to loss of changes that were made since the last transaction commit. This may be up to 30 seconds (default commit period) or less if the commit was implied by other filesystem activity.

One can determine whether **zero−log** is needed according to the kernel backtrace:

```
? replay_one_dir_item+0xb5/0xb5 [btrfs]
? walk_log_tree+0x9c/0x19d [btrfs]
? btrfs_read_fs_root_no_radix+0x169/0x1a1 [btrfs]
? btrfs_recover_log_trees+0x195/0x29c [btrfs]
? replay_one_dir_item+0xb5/0xb5 [btrfs]
? btree_read_extent_buffer_pages+0x76/0xbc [btrfs]
? open_ctree+0xff6/0x132c [btrfs]
```

If the errors are like above, then **zero−log** should be used to clear the log and the filesystem may be mounted normally again. The keywords to look for are 'open_ctree' which says that it's during mount and function names that contain *replay*, *recover* or *log_tree*.

EXIT STATUS

btrfs rescue 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−check(8) <>, btrfs−scrub(8) <>, mkfs.btrfs(8) <>