NAME

btrfs-subvolume − manage btrfs subvolumes

SYNOPSIS

btrfs subvolume <*subcommand*> [<*args*>]

DESCRIPTION

btrfs subvolume is used to create/delete/list/show btrfs subvolumes and snapshots.

SUBVOLUME AND SNAPSHOT

A subvolume is a part of filesystem with its own independent file/directory hierarchy. Subvolumes can share file extents. A snapshot is also subvolume, but with a given initial content of the original subvolume.

Note

A subvolume in btrfs is not like an LVM logical volume, which is block−level snapshot while btrfs subvolumes are file extent−based.

A subvolume looks like a normal directory, with some additional operations described below. Subvolumes can be renamed or moved, nesting subvolumes is not restricted but has some implications regarding snapshotting.

A subvolume in btrfs can be accessed in two ways:

- like any other directory that is accessible to the user
- like a separately mounted filesystem (options *subvol* or *subvolid*)

In the latter case the parent directory is not visible and accessible. This is similar to a bind mount, and in fact the subvolume mount does exactly that.

A freshly created filesystem is also a subvolume, called *top−level*, internally has an id 5. This subvolume cannot be removed or replaced by another subvolume. This is also the subvolume that will be mounted by default, unless the default subvolume has been changed (see subcommand *set−default*).

A snapshot is a subvolume like any other, with given initial content. By default, snapshots are created read−write. File modifications in a snapshot do not affect the files in the original subvolume.

SUBCOMMAND

create [−i <*qgroupid*>] [<dest>/]<*name*>

Create a subvolume <*name*> in <*dest*>.

If <*dest*> is not given, subvolume <*name*> will be created in the current directory.

Options

−i <*qgroupid*>

Add the newly created subvolume to a qgroup. This option can be given multiple times.

delete [options] <*[*<*subvolume*> [<*subvolume*>...]], **delete** −i|−−subvolid <*subvolid*> <*path*>>

Delete the subvolume(s) from the filesystem.

If <*subvolume*> is not a subvolume, btrfs returns an error but continues if there are more arguments to process.

If −−subvolid is used, <*path*> must point to a btrfs filesystem. See **btrfs subvolume list** or **btrfs inspect−internal rootid** how to get the subvolume id.

The corresponding directory is removed instantly but the data blocks are removed later in the background. The command returns immediately. See **btrfs subvolume sync** how to wait until the subvolume gets completely removed.

The deletion does not involve full transaction commit by default due to performance reasons. As a consequence, the subvolume may appear again after a crash. Use one of the −−*commit* options to wait until the operation is safely stored on the device.

The default subvolume (see **btrfs subvolume set−default**) cannot be deleted and returns error (EPERM) and this is logged to the system log. A subvolume that's currently involved in send (see **btrfs send**) also cannot be deleted until the send is finished. This is also logged in the system log.

Options

−c|−−commit−after
wait for transaction commit at the end of the operation.

−C|−−commit−each
wait for transaction commit after deleting each subvolume.

−i|−−subvolid <*subvolid*>
subvolume id to be removed instead of the <*path*> that should point to the filesystem with the subvolume

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

find−new <*subvolume*> <*last_gen*>
List the recently modified files in a subvolume, after <*last_gen*> generation.

get−default <*path*>
Get the default subvolume of the filesystem <*path*>.

The output format is similar to **subvolume list** command.

list [options] [−G [+|−]<*value*>] [−C [+|−]<*value*>] [−−sort=rootid,gen,ogen,path] <*path*>
List the subvolumes present in the filesystem <*path*>.

For every subvolume the following information is shown by default:

ID <*ID*> gen <*generation*> top level <*ID*> path <*path*>

where ID is subvolume's id, gen is an internal counter which is updated every transaction, top level is the same as parent subvolume's id, and path is the relative path of the subvolume to the top level subvolume. The subvolume's ID may be used by the subvolume set−default command, or at mount time via the subvolid= option.

Options

Path filtering

−o
print only subvolumes below specified <*path*>.

−a
print all the subvolumes in the filesystem and distinguish between absolute and relative path with respect to the given <*path*>.

Field selection

−p
print the parent ID (*parent* here means the subvolume which contains this subvolume).

−c
print the ogeneration of the subvolume, aliases: ogen or origin generation.

−g

print the generation of the subvolume (default).

−u
print the UUID of the subvolume.

−q
print the parent UUID of the subvolume (*parent* here means subvolume of which this subvolume is a snapshot).

−R
print the UUID of the sent subvolume, where the subvolume is the result of a receive operation.

Type filtering

−s
only snapshot subvolumes in the filesystem will be listed.

−r
only readonly subvolumes in the filesystem will be listed.

−d
list deleted subvolumes that are not yet cleaned.

Other

−t
print the result as a table.

Sorting
By default the subvolumes will be sorted by subvolume ID ascending.

−G [+|−]<*value*>
list subvolumes in the filesystem that its generation is >=, 〈 or = value. '+' means >= value, '−' means <= value, If there is neither '+' nor '−', it means = value.

−C [+|−]<*value*>
list subvolumes in the filesystem that its ogeneration is >=, <= or = value. The usage is the same to *−G* option.

−−sort=rootid,gen,ogen,path
list subvolumes in order by specified items. you can add '+' or '−' in front of each items, '+' means ascending, '−' means descending. The default is ascending.

for −−sort you can combine some items together by ',', just like −−sort=+ogen,−gen,path,rootid.

set−default [<*subvolume*>|<*id*> <*path*>]
Set the default subvolume for the (mounted) filesystem.

Set the default subvolume for the (mounted) filesystem at <*path*>. This will hide the top−level subvolume (i.e. the one mounted with *subvol=/* or *subvolid=5*). Takes action on next mount.

There are two ways how to specify the subvolume, by <*id*> or by the <*subvolume*> path. The id can be obtained from **btrfs subvolume list**, **btrfs subvolume show** or **btrfs inspect−internal rootid**.

show [options] <*path*>
Show more information about a subvolume (UUIDs, generations, times, flags, related snapshots).

/mnt/btrfs/subvolume
Name: subvolume
UUID: 5e076a14−4e42−254d−ac8e−55bebea982d1
Parent UUID: −

```
Received UUID:        −
Creation time:        2018−01−01 12:34:56 +0000
Subvolume ID:         79
Generation:           2844
Gen at creation:      2844
Parent ID:            5
Top level ID:         5
Flags:                −
Snapshot(s):
```

Options

−r|−−rootid <*ID*>
: show details about subvolume with root <*ID*>, looked up in <*path*>

−u|−−uuid UUID
: show details about subvolume with the given <*UUID*>, looked up in <*path*>

snapshot [−r] [−i <*qgroupid*>] <*source*> <*dest*>|[<dest>/]<*name*>
: Create a snapshot of the subvolume <*source*> with the name <*name*> in the <*dest*> directory.

  If only <*dest*> is given, the subvolume will be named the basename of <*source*>. If <*source*> is not a subvolume, btrfs returns an error.

  Options

  −r
  : Make the new snapshot read only.

  −i <*qgroupid*>
  : Add the newly created subvolume to a qgroup. This option can be given multiple times.

sync <*path*> [subvolid...]
: Wait until given subvolume(s) are completely removed from the filesystem after deletion. If no subvolume id is given, wait until all current deletion requests are completed, but do not wait for subvolumes deleted in the meantime.

  Options

  −s <*N*>
  : sleep N seconds between checks (default: 1)

SUBVOLUME FLAGS

The subvolume flag currently implemented is the *ro* property. Read−write subvolumes have that set to *false*, snapshots as *true*. In addition to that, a plain snapshot will also have last change generation and creation generation equal.

Read−only snapshots are building blocks fo incremental send (see **btrfs−send**(8)) and the whole use case relies on unmodified snapshots where the relative changes are generated from. Thus, changing the subvolume flags from read−only to read−write will break the assumptions and may lead to unexpected changes in the resulting incremental stream.

A snapshot that was created by send/receive will be read−only, with different last change generation, read−only and with set *received_uuid* which identifies the subvolume on the filesystem that produced the stream. The usecase relies on matching data on both sides. Changing the subvolume to read−write after it has been received requires to reset the *received_uuid*. As this is a notable change and could potentially break the incremental send use case, performing it by **btrfs property set** requires force if that is really desired by user.

Note

The safety checks have been implemented in 5.14.2, any subvolumes previously received (with a valid *received_uuid*) and read−write status may exist and could still lead to problems with send/receive. You can use **btrfs subvolume show** to identify them. Flipping the flags to read−only and back to read−write will reset the *received_uuid* manually. There may exist a convenience tool in the future.

EXAMPLES

Example 1. Deleting a subvolume

If we want to delete a subvolume called **foo** from a btrfs volume mounted at **/mnt/bar** we could run the following:

```
btrfs subvolume delete /mnt/bar/foo
```

EXIT STATUS

btrfs subvolume returns a zero exit status if it succeeds. A non−zero value is returned in case of failure.

AVAILABILITY

btrfs is part of btrfs−progs. Please refer to the btrfs wiki **http://btrfs.wiki.kernel.org** for further details.

SEE ALSO

mkfs.btrfs(8), **mount**(8), **btrfs−quota**(8), **btrfs−qgroup**(8), **btrfs−send**(8)