



Linux Ubuntu 26.04 Manual Pages on command 'btrfs-ioctl.2'

\$ man btrfs-ioctl.2

BTRFS-IOCTL(2) BTRFS BTRFS-IOCTL(2)

NAME

btrfs-ioctl - documentation about btrfs ioctls

NAME

btrfs-ioctl - documentation for the ioctl interface to btrfs

DESCRIPTION

The ioctl() system call is a way how to request custom actions performed on a filesystem beyond the standard interfaces (like syscalls). An ioctl is specified by a number and an associated data structure that implement a feature, usually not available in other filesystems. The number of ioctls grows over time and in some cases get promoted to a VFS-level ioctl once other filesystems adopt the functionality. Backward compatibility is maintained and a formerly private ioctl number could become available on the VFS level.

OVERVIEW

The ioctls are defined by a number and associated with a data structure that contains further information. All ioctls use file descriptor (fd) as a reference point, it could be the filesystem or a directory inside the filesystem.

An ioctl can be used in the following schematic way:

```
struct btrfs_ioctl_args args;
memset(&args, 0, sizeof(args));
args.key = value;
ret = ioctl(fd, BTRFS_IOC_NUMBER, &args);
```

The fd is the entry point to the filesystem and for most ioctls it does not matter which directory is that. A distinction between files and directories sometimes matter, when it matters.

ters it's explicitly mentioned. The args is the associated data structure for the request. It's strongly recommended to initialize the whole structure to zeros as this is future-proof when the ioctl gets further extensions. Not doing that could lead to mismatch of old user space and new kernel versions, or vice versa. The BTRFS_IOC_NUMBER is says which operation should be done on the given arguments. Some ioctls take a specific data structure, some of them share a common one, no argument structure ioctls exist too. The data passed to an ioctl can be input, output or both.

The library libbtrfsutil wraps a few ioctls for convenience. Using raw ioctls is not discouraged but may be cumbersome though it does not need additional library dependency. Backward compatibility is guaranteed and incompatible changes usually lead to a new version of the ioctl. Enhancements of existing ioctls can happen and depend on additional flags to be set. Zeroed unused space is commonly understood as a mechanism to communicate the compatibility between kernel and userspace and thus zeroing is really important. In exceptional cases this is not enough and further flags need to be passed to distinguish between zero as implicit unused initialization and a valid zero value. Such cases are documented.

File descriptors of regular files are obtained by `int fd = open()`, directories opened as `DIR *dir = opendir()` can be converted to the corresponding file descriptor by `fd = dirfd(dir)`.

LIST OF IOCTLS

??			
? Name	? Description	? Data	?
??			
? BTRFS_IOC_SNAP_CREATE	? (obsolete) create a snap?	? struct	?
?	? shot of a subvolume	? btrfs_ioctl_vol_args	?
??			
? BTRFS_IOC_DEFRAG	?	?	?
??			
? BTRFS_IOC_RESIZE	?	?	?
??			
? BTRFS_IOC_SCAN_DEV	? scan and register a given	? struct	?
?	? device path with filesystem?	? btrfs_ioctl_vol_args	?
?	? tem module	?	?
??			
? BTRFS_IOC_SYNC	? Sync the filesystem, pos?	? NULL	?

? ? sibly process queued up ? ?

? ? work ? ?

??

? BTRFS_IOC_CLONE ? ? ?

??

? BTRFS_IOC_ADD_DEV ? add a device to the ? struct ?

? ? filesystem by path ? btrfs_ioctl_vol_args ?

??

? BTRFS_IOC_RM_DEV ? delete a device from the ? struct ?

? ? filesystem by path ? btrfs_ioctl_vol_args ?

??

? BTRFS_IOC_BALANCE ? ? ?

??

? BTRFS_IOC_CLONE_RANGE ? ? ?

??

? BTRFS_IOC_SUBVOL_CREATE ? (obsolete) create a sub? ? struct ?

? ? volume ? btrfs_ioctl_vol_args ?

??

? BTRFS_IOC_SNAP_DESTROY ? (obsolete) delete a sub? ? struct ?

? ? volume ? btrfs_ioctl_vol_args ?

??

? BTRFS_IOC_DEFRAG_RANGE ? ? ?

??

? BTRFS_IOC_TREE_SEARCH ? ? ?

??

? BTRFS_IOC_TREE_SEARCH_V2 ? ? ?

??

? BTRFS_IOC_INO_LOOKUP ? resolve inode number to ? struct ?

? ? path, or lookup contain? ? btrfs_ioctl_ino_lookup_args ?

? ? ing subvolume id ? ?

??

? BTRFS_IOC_DEFAULT_SUBVOL ? set the default subvolume ? uint64_t ?

? ? id ? ?

??
? BTRFS_IOC_SPACE_INFO ? ? ?
??
? BTRFS_IOC_START_SYNC ? ? ?
??
? BTRFS_IOC_WAIT_SYNC ? ? ?
??
? BTRFS_IOC_SNAP_CREATE_V2 ? create a snapshot of a ? struct ?
? ? subvolume ? btrfs_ioctl_vol_args_v2 ?
??
? BTRFS_IOC_SUBVOL_CREATE ? create a subvolume ? struct ?
? ATE_V2 ? ? btrfs_ioctl_vol_args_v2 ?
??
? BTRFS_IOC_SUBVOL_GETFLAGS ? get flags of a subvolume ? uint64_t ?
??
? BTRFS_IOC_SUBVOL_SETFLAGS ? set flags of a subvolume ? uint64_t ?
??
? BTRFS_IOC_SCRUB ? ? ?
??
? BTRFS_IOC_SCRUB_CANCEL ? ? ?
??
? BTRFS_IOC_SCRUB_PROGRESS ? ? ?
??
? BTRFS_IOC_DEV_INFO ? get information about a ? struct ?
? ? device (UUIDs, used size, ? btrfs_ioctl_dev_info_args ?
? ? total size) ? ?
??
? BTRFS_IOC_FS_INFO ? get information about ? struct ?
? ? filesystem (device count, ? btrfs_ioctl_fs_info_args ?
? ? fsid, ...) ? ?
??
? BTRFS_IOC_BALANCE_V2 ? ? ?
??

? BTRFS_IOC_BALANCE_CTL ? ?
??
? BTRFS_IOC_BAL? ? ?
? ANCE_PROGRESS ? ?
??
? BTRFS_IOC_INO_PATHS ? ?
??
? BTRFS_IOC_LOGICAL_INO ? ?
??
? BTRFS_IOC_SET_RE? ? ?
? CEIVED_SUBVOL ? ?
??
? BTRFS_IOC_SEND ? ?
??
? BTRFS_IOC_DEVICES_READY ? ?
??
? BTRFS_IOC_QUOTA_CTL ? ?
??
? BTRFS_IOC_QGROUP_ASSIGN ? ?
??
? BTRFS_IOC_QGROUP_CREATE ? ?
??
? BTRFS_IOC_QGROUP_LIMIT ? ?
??
? BTRFS_IOC_QUOTA_RESCAN ? ?
??
? BTRFS_IOC_QUOTA_RES? ? ?
? CAN_STATUS ? ?
??
? BTRFS_IOC_QUOTA_RES? ? ?
? CAN_WAIT ? ?
??
? BTRFS_IOC_GET_FSLABEL ? read filesystem label ? char buffer[BTRFS_LA? ?

? ? ? BEL_SIZE] ?
??
? BTRFS_IOC_SET_FSLABEL ? set the filesystem label ? char buffer[BTRFS_LA? ?
? ? ? BEL_SIZE] ?
??
? BTRFS_IOC_GET_DEV_STATS ? ? ?
??
? BTRFS_IOC_DEV_REPLACE ? ? ?
??
? BTRFS_IOC_FILE_EX? ? ? ?
? TENT_SAME ? ? ?
??
? BTRFS_IOC_GET_FEATURES ? get features set on the ? struct btrfs_ioctl_fea? ?
? ? filesystem ? ture_flags ?
??
? BTRFS_IOC_SET_FEATURES ? set features on the ? struct btrfs_ioctl_fea? ?
? ? filesystem ? ture_flags ?
??
? BTRFS_IOC_GET_SUP? ? get available filesystem ? struct btrfs_ioctl_fea? ?
? PORTED_FEATURES ? feature sets ? ture_flags[3] ?
??
? BTRFS_IOC_RM_DEV_V2 ? ? ?
??
? BTRFS_IOC_LOGICAL_INO_V2 ? ? ?
??
? BTRFS_IOC_GET_SUBVOL_INFO ? get information about a ? struct btrfs_ioctl_get_sub? ?
? ? subvolume ? vol_info_args ?
??
? BTRFS_IOC_GET_SUB? ? ? ?
? VOL_ROOTREF ? ? ?
??
? BTRFS_IOC_INO_LOOKUP_USER ? ? ?
??

? BTRFS_IOC_SNAP_DESTROY_V2 ? destroy a (snapshot or ? struct ?
 ? ? regular) subvolume ? btrfs_ioctl_vol_args_v2 ?
 ???
 ? BTRFS_IOC_ENCODED_READ ? ? ?
 ???
 ? BTRFS_IOC_ENCODED_WRITE ? ? ?
 ???
 ? BTRFS_IOC_SUB? ? Wait until a deleted sub? ? struct btrfs_ioctl_sub? ?
 ? VOL_SYNC_WAIT ? volume is cleaned or ? vol_wait ?
 ? ? query the state. ? ?
 ???

DATA STRUCTURES AND DEFINITIONS

```
struct btrfs_ioctl_vol_args {
    __s64 fd;

    char name[BTRFS_PATH_NAME_MAX + 1];
};

#define BTRFS_SUBVOL_RDONLY (1ULL << 1)
#define BTRFS_SUBVOL_QGROUP_INHERIT (1ULL << 2)
#define BTRFS_DEVICE_SPEC_BY_ID (1ULL << 3)
#define BTRFS_SUBVOL_SPEC_BY_ID (1ULL << 4)

struct btrfs_ioctl_vol_args_v2 {
    __s64 fd;
    __u64 transid;
    __u64 flags;
    union {
        struct {
            __u64 size;

            struct btrfs_qgroup_inherit __user *qgroup_inherit;
        };
        __u64 unused[4];
    };
    union {
        char name[BTRFS_SUBVOL_NAME_MAX + 1];
```

```

    __u64 devid;

    __u64 subvalid;

};

};

#define BTRFS_FEATURE_COMPAT_RO_FREE_SPACE_TREE    (1ULL << 0)

/*
 * Older kernels (< 4.9) on big-endian systems produced broken free space tree
 * bitmaps, and btrfs-progs also used to corrupt the free space tree (versions
 * < 4.7.3). If this bit is clear, then the free space tree cannot be trusted.
 * btrfs-progs can also intentionally clear this bit to ask the kernel to
 * rebuild the free space tree, however this might not work on older kernels
 * that do not know about this bit. If not sure, clear the cache manually on
 * first mount when booting older kernel versions.
 */

#define BTRFS_FEATURE_COMPAT_RO_FREE_SPACE_TREE_VALID (1ULL << 1)
#define BTRFS_FEATURE_COMPAT_RO_VERITY                (1ULL << 2)
#define BTRFS_FEATURE_COMPAT_RO_BLOCK_GROUP_TREE     (1ULL << 3)
#define BTRFS_FEATURE_INCOMPAT_MIXED_BACKREF         (1ULL << 0)
#define BTRFS_FEATURE_INCOMPAT_DEFAULT_SUBVOL        (1ULL << 1)
#define BTRFS_FEATURE_INCOMPAT_MIXED_GROUPS          (1ULL << 2)
#define BTRFS_FEATURE_INCOMPAT_COMPRESS_LZO          (1ULL << 3)
#define BTRFS_FEATURE_INCOMPAT_COMPRESS_ZSTD         (1ULL << 4)
#define BTRFS_FEATURE_INCOMPAT_BIG_METADATA          (1ULL << 5)
#define BTRFS_FEATURE_INCOMPAT_EXTENDED_IREF         (1ULL << 6)
#define BTRFS_FEATURE_INCOMPAT_RAID56                (1ULL << 7)
#define BTRFS_FEATURE_INCOMPAT_SKINNY_METADATA       (1ULL << 8)
#define BTRFS_FEATURE_INCOMPAT_NO_HOLES              (1ULL << 9)
#define BTRFS_FEATURE_INCOMPAT_METADATA_UUID         (1ULL << 10)
#define BTRFS_FEATURE_INCOMPAT_RAID1C34              (1ULL << 11)
#define BTRFS_FEATURE_INCOMPAT_ZONED                 (1ULL << 12)
#define BTRFS_FEATURE_INCOMPAT_EXTENT_TREE_V2        (1ULL << 13)
#define BTRFS_FEATURE_INCOMPAT_RAID_STRIPE_TREE      (1ULL << 14)
#define BTRFS_FEATURE_INCOMPAT_SIMPLE_QUOTA          (1ULL << 16)

```



```

struct btrfs_ioctl_feature_flags {
    __u64 compat_flags;
    __u64 compat_ro_flags;
    __u64 incompat_flags;
};

struct btrfs_ioctl_get_subvol_info_args {
    /* Id of this subvolume */
    __u64 treeid;

    /* Name of this subvolume, used to get the real name at mount point */
    char name[BTRFS_VOL_NAME_MAX + 1];

    /*
     * Id of the subvolume which contains this subvolume.
     * Zero for top-level subvolume or a deleted subvolume.
     */
    __u64 parent_id;

    /*
     * Inode number of the directory which contains this subvolume.
     * Zero for top-level subvolume or a deleted subvolume
     */
    __u64 dirid;

    /* Latest transaction id of this subvolume */
    __u64 generation;

    /* Flags of this subvolume */
    __u64 flags;

    /* UUID of this subvolume */
    __u8 uuid[BTRFS_UUID_SIZE];

    /*
     * UUID of the subvolume of which this subvolume is a snapshot.
     * All zero for a non-snapshot subvolume.
     */
    __u8 parent_uuid[BTRFS_UUID_SIZE];

    /*
     * UUID of the subvolume from which this subvolume was received.

```

```

* All zero for non-received subvolume.

*/

__u8 received_uuid[BTRFS_UUID_SIZE];

/* Transaction id indicating when change/create/send/receive happened */

__u64 ctransid;

__u64 otransid;

__u64 stransid;

__u64 rtransid;

/* Time corresponding to c/o/s/rtransid */

struct btrfs_ioctl_timespec ctime;

struct btrfs_ioctl_timespec otime;

struct btrfs_ioctl_timespec stime;

struct btrfs_ioctl_timespec rtime;

/* Must be zero */

__u64 reserved[8];

};

#define BTRFS_QGROUP_INHERIT_SET_LIMITS (1ULL << 0)

struct btrfs_qgroup_inherit {

    __u64 flags;

    __u64 num_qgroups;

    __u64 num_ref_copies;

    __u64 num_excl_copies;

    struct btrfs_qgroup_limit lim;

    __u64 qgroups[];

};

#define BTRFS_QGROUP_LIMIT_MAX_RFER (1ULL << 0)

#define BTRFS_QGROUP_LIMIT_MAX_EXCL (1ULL << 1)

#define BTRFS_QGROUP_LIMIT_RSV_RFER (1ULL << 2)

#define BTRFS_QGROUP_LIMIT_RSV_EXCL (1ULL << 3)

#define BTRFS_QGROUP_LIMIT_RFER_CMPR (1ULL << 4)

#define BTRFS_QGROUP_LIMIT_EXCL_CMPR (1ULL << 5)

struct btrfs_qgroup_limit {

    __u64 flags;

```

```

    __u64 max_rfer;

    __u64 max_excl;

    __u64 rsv_rfer;

    __u64 rsv_excl;

};

struct btrfs_ioctl_dev_info_args {
    __u64 devid;                /* in/out */
    __u8 uuid[BTRFS_UUID_SIZE]; /* in/out */
    __u64 bytes_used;           /* out */
    __u64 total_bytes;          /* out */
    /*
     * Optional, out.
     *
     * Showing the fsid of the device, allowing user space to check if this
     * device is a seeding one.
     *
     * Introduced in v6.3, thus user space still needs to check if kernel
     * changed this value. Older kernel will not touch the values here.
     */
    __u8 fsid[BTRFS_UUID_SIZE];
    __u64 unused[377];          /* pad to 4k */
    __u8 path[BTRFS_DEVICE_PATH_NAME_MAX]; /* out */
};

/* Request information about checksum type and size */
#define BTRFS_FS_INFO_FLAG_CSUM_INFO (1U << 0)

/* Request information about filesystem generation */
#define BTRFS_FS_INFO_FLAG_GENERATION (1U << 1)

/* Request information about filesystem metadata UUID */
#define BTRFS_FS_INFO_FLAG_METADATA_UUID (1U << 2)

struct btrfs_ioctl_fs_info_args {
    __u64 max_id;                /* out */
    __u64 num_devices;           /* out */
    __u8 fsid[BTRFS_FSID_SIZE]; /* out */

```

```

    __u32 nodesize;          /* out */
    __u32 sectorsize;        /* out */
    __u32 clone_alignment;    /* out */
    /* See BTRFS_FS_INFO_FLAG_ */
    __u16 csum_type;          /* out */
    __u16 csum_size;          /* out */
    __u64 flags;              /* in/out */
    __u64 generation;         /* out */
    __u8 metadata_uuid[BTRFS_FSID_SIZE]; /* out */
    __u8 reserved[944];       /* pad to 1k */
};

#define BTRFS_INO_LOOKUP_PATH_MAX    4080

struct btrfs_ioctl_ino_lookup_args {
    __u64 treeid;
    __u64 objectid;
    char name[BTRFS_INO_LOOKUP_PATH_MAX];
};

/* Specify the subvol. */

#define BTRFS_SUBVOL_SYNC_WAIT_FOR_ONE    (0)

/* Wait for all currently queued. */

#define BTRFS_SUBVOL_SYNC_WAIT_FOR_QUEUED    (1)

/* Count number of queued subvolumes. */

#define BTRFS_SUBVOL_SYNC_COUNT    (2)

/*
 * Read which is the first in the queue (to be cleaned or being cleaned already),
 * or 0 if the queue is empty.
 */

#define BTRFS_SUBVOL_SYNC_PEEK_FIRST    (3)

/* Read the last subvol in the queue, or 0 if the queue is empty. */

#define BTRFS_SUBVOL_SYNC_PEEK_LAST    (4)

struct btrfs_ioctl_subvol_wait {
    __u64 subvol;
    __u32 mode;

```

__u32 count;

};

??

? Constant name ? Value ?

??

? BTRFS_UUID_SIZE ? 16 ?

??

? BTRFS_FSID_SIZE ? 16 ?

??

? BTRFS_SUBVOL_NAME_MAX ? 4039 ?

??

? BTRFS_PATH_NAME_MAX ? 4087 ?

??

? BTRFS_VOL_NAME_MAX ? 255 ?

??

? BTRFS_LABEL_SIZE ? 256 ?

??

? BTRFS_FIRST_FREE_OBJECTID ? 256 ?

??

DETAILED DESCRIPTION

BTRFS_IOC_SNAP_CREATE

Note:

obsoleted by BTRFS_IOC_SNAP_CREATE_V2

(since: 3.0, obsoleted: 4.0) Create a snapshot of a subvolume.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of the parent di? ?

? ? rectory of the new subvolume ?

??

? ioctl args ? struct btrfs_ioctl_vol_args ?

??

? args.fd ? file descriptor of any directory ?

? ? inside the subvolume to snapshot, ?

? ? must be on the same filesystem ?

??

? args.name ? name of the subvolume, although ?

? ? the buffer can be almost 4KiB, ?

? ? the file size is limited by Linux ?

? ? VFS to 255 characters and must ?

? ? not contain a slash ('/') ?

??

BTRFS_IOC_SCAN_DEV

Scan and register a given device in the filesystem module, which can be later used for automatic device and filesystem association at mount time. This operates on the control device, not files from a mounted filesystem. Can be safely called repeatedly with same device path.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of the control ?

? ? device /dev/btrfs-control ?

??

? ioctl args ? struct btrfs_ioctl_vol_args ?

??

? args.fd ? ignored ?

??

? args.name ? full path of the device ?

??

BTRFS_IOC_SYNC

Sync the filesystem data as would sync() syscall do, additionally wake up the internal transaction thread that may trigger actions like subvolume cleaning or queued defragmentation.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of any file or ?

? ? directory in the filesystem ?
??
? ioctl args ? NULL ?
??

BTRFS_IOC_ADD_DEV

Add a given block device to the filesystem. Unlike the command btrfs device add there's are no safety checks (like existence of another filesystem on the device), device preparation (like TRIM or zone reset), so use it with care.

This is a filesystem-exclusive operation and it will fail if there's another one already running, with one exception, when there's a paused balance.

Required permissions: CAP_SYS_ADMIN

??
? Field ? Description ?
??
? ioctl fd ? file descriptor of any file or ?
? ? directory in the filesystem ?
??
? ioctl args ? struct btrfs_ioctl_vol_args ?
??
? args.fd ? ignored ?
??
? args.name ? full path of the block device to ?
? ? be added ?
??

BTRFS_IOC_RM_DEV

Remove a device from the filesystem specified by it's path, or cancel a running device deletion by special path cancel.

This is a filesystem-exclusive operation and it will fail if there's another one already running.

Required permissions: CAP_SYS_ADMIN

??
? Field ? Description ?
??

? ioctl fd ? file descriptor of any file or ?

? ? directory in the filesystem ?

??

? ioctl args ? struct btrfs_ioctl_vol_args ?

??

? args.fd ? ignored ?

??

? args.name ? full path of the block device to ?

? ? be deleted or string "cancel" ?

??

BTRFS_IOC_SUBVOL_CREATE

Note:

obsoleted by BTRFS_IOC_SUBVOL_CREATE_V2

(since: 3.0, obsoleted: 4.0) Create a subvolume.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of the parent di? ?

? ? rectory of the new subvolume ?

??

? ioctl args ? struct btrfs_ioctl_vol_args ?

??

? args.fd ? ignored ?

??

? args.name ? name of the subvolume, although ?

? ? the buffer can be almost 4KiB, ?

? ? the file size is limited by Linux ?

? ? VFS to 255 characters and must ?

? ? not contain a slash ("/) ?

??

BTRFS_IOC_SNAP_DESTROY

Note:

obsoleted by BTRFS_IOC_SNAP_DESTROY_V2

(since: 2.6.33, obsoleted: 5.7) Delete a subvolume.

```
????????????????????????????????????????????????????????
? Field      ? Description              ?
????????????????????????????????????????????????????????
? ioctl fd   ? file descriptor of the parent di? ?
?           ? rectory of the new subvolume    ?
????????????????????????????????????????????????????????
? ioctl args ? struct btrfs_ioctl_vol_args    ?
????????????????????????????????????????????????????????
? args.fd    ? ignored                    ?
????????????????????????????????????????????????????????
? args.name  ? name of the subvolume, although ?
?           ? the buffer can be almost 4KiB, ?
?           ? the file size is limited by Linux ?
?           ? VFS to 255 characters and must ?
?           ? not contain a slash ('/')      ?
????????????????????????????????????????????????????????
```

BTRFS_IOC_INO_LOOKUP

Resolve inode number to a path (requires CAP_SYS_ADMIN), or read a containing subvolume id of the given file (unrestricted, special case). The size of the path name buffer is shorter than PATH_MAX (4096), it's possible that the path is trimmed due to that. Also implemented by btrfs inspect-internal rootid <#man-inspect-rootid>.

The general case needs CAP_SYS_ADMIN and can resolve any file to its path. The special case for reading the containing subvolume is not restricted:

```
struct btrfs_ioctl_ino_lookup_args args;

fd = open("file", ...);

args.treeid = 0;

args.objectid = BTRFS_FIRST_FREE_OBJECTID;

ioctl(fd, BTRFS_IOC_INO_LOOKUP, &args);

/* args.treeid now contains the subvolume id */

????????????????????????????????????????????????????????
? Field      ? Description              ?
????????????????????????????????????????????????????????
```

? ioctl fd ? file descriptor of the file or ?

? ? directory to lookup the subvol? ?

? ? umeid ?

??

? ioctl args ? struct ?

? ? btrfs_ioctl_ino_lookup_args ?

??

? args.treeid ? subvolume id against which the ?

? ? path should be resolved (needs ?

? ? CAP_SYS_ADMIN), or 0 so the sub? ?

? ? volume containing fd will be used ?

??

? args.objectid ? inode number to lookup, IN? ?

? ? ODE_REF_KEY with that key.objec? ?

? ? tid, or BTRFS_FIRST_FREE_OBJECTID ?

? ? as special case to read only the ?

? ? tree id and clear the args.name ?

? ? buffer ?

??

? args.name ? path relative to the toplevel ?

? ? subvolume, or empty string ?

??

BTRFS_IOC_DEFAULT_SUBVOL

Set the given subvolume id as the default one when mounting the filesystem without sub?

vol=path or subvolid=id options.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of the directory ?

? ? inside which to create the new ?

? ? snapshot ?

??

? ioctl args ? numeric value of subvolume to be? ?

? ? come default (uint64_t) ?
??

BTRFS_IOC_SNAP_CREATE_V2

Create a snapshot of a subvolume.

??
? Field ? Description ?
??
? ioctl fd ? file descriptor of the directory ?
? ? inside which to create the new ?
? ? snapshot ?
??
? ioctl args ? struct btrfs_ioctl_vol_args_v2 ?
??
? args.fd ? file descriptor of any directory ?
? ? inside the subvolume to snapshot, ?
? ? must be on the filesystem ?
??
? args.transid ? ignored ?
??
? args.flags ? any subset of BTRFS_SUBVOL_RDONLY ?
? ? to make the new snapshot ?
? ? read-only, or BTRFS_SUBVOL_RDONLY ?
? ? VOL_QGROUP_INHERIT to apply the ?
? ? qgroup_inherit field ?
??
? args.name ? the name, under the ioctl fd, for ?
? ? the new subvolume ?
??

BTRFS_IOC_SUBVOL_CREATE_V2

(since: 3.6) Create a subvolume, qgroup inheritance and limits can be specified.

??
? Field ? Description ?
??

? ioctl fd ? file descriptor of the parent di? ?

? ? rectory of the new subvolume ?

??

? ioctl args ? struct btrfs_ioctl_vol_args_v2 ?

??

? args.fd ? ignored ?

??

? args.transid ? ignored ?

??

? args.flags ? flags to set on the subvolume, ?

? ? BTRFS_SUBVOL_RDONLY for readonly, ?

? ? BTRFS_SUBVOL_QGROUP_INHERIT if ?

? ? the qgroup related fields should ?

? ? be processed ?

??

? args.size ? number of entries in ?

? ? args.qgroup_inherit ?

??

? args.qgroup_inherit ? inherit the given qgroups (struct ?

? ? btrfs_qgroup_inherit) and limits ?

? ? (struct btrfs_qgroup_limit) ?

??

? name ? name of the subvolume, although ?

? ? the buffer can be almost 4KiB, ?

? ? the file size is limited by Linux ?

? ? VFS to 255 characters and must ?

? ? not contain a slash ('/') ?

??

BTRFS_IOC_SUBVOL_GETFLAGS

Read the flags of a subvolume. The returned flags are either 0 or BTRFS_SUBVOL_RDONLY.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of the subvolume ?

? ? to examine ?

??

? ioctl args ? uint64_t ?

??

BTRFS_IOC_SUBVOL_SETFLAGS

Change the flags of a subvolume.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of the subvolume ?

? ? to modify ?

??

? ioctl args ? uint64_t, either 0 or BTRFS_SUB? ?

? ? VOL_RDONLY ?

??

BTRFS_IOC_GET_FSLABEL

Read the label of the filesystem into a given buffer. Alternatively it can be read from /sys/fs/btrfs/FSID/label though it requires to know the FSID of the filesystem.

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of any file/di? ?

? ? rectory in the filesystem ?

??

? ioctl args ? char buffer[BTRFS_LABEL_SIZE] ?

??

BTRFS_IOC_SET_FSLABEL

Set the label of filesystem from given buffer. The maximum length also accounts for terminating NUL character. Alternatively it can be also set by writing to /sys/fs/btrfs/FSID/label though it requires to know the FSID of the filesystem (and an explicit commit before the change is permanent).

```

????????????????????????????????????????????????????????
? Field    ? Description          ?
????????????????????????????????????????????????????????
? ioctl fd  ? file descriptor of any file/di? ?
?          ? rectory in the filesystem      ?
????????????????????????????????????????????????????????
? ioctl args ? char buffer[BTRFS_LABEL_SIZE] ?
????????????????????????????????????????????????????????

```

BTRFS_IOC_DEV_INFO

Read some basic information about a device, requested by the devid or device UUID.

```

????????????????????????????????????????????????????????
? Field    ? Description          ?
????????????????????????????????????????????????????????
? ioctl fd  ? file descriptor of any file/di? ?
?          ? rectory in the filesystem      ?
????????????????????????????????????????????????????????
? ioctl args ? struct btrfs_ioctl_dev_info_args ?
????????????????????????????????????????????????????????

```

BTRFS_IOC_FS_INFO

Read internal information about the filesystem. The data can be exchanged both ways and part of the structure could be optionally filled. The reserved bytes can be used to get new kind of information in the future, always depending on the flags set.

```

????????????????????????????????????????????????????????
? Field    ? Description          ?
????????????????????????????????????????????????????????
? ioctl fd  ? file descriptor of any file/di? ?
?          ? rectory in the filesystem      ?
????????????????????????????????????????????????????????
? ioctl args ? struct btrfs_ioctl_fs_info_args ?
????????????????????????????????????????????????????????

```

BTRFS_IOC_GET_FEATURES

Get the actually set feature bits on the filesystem (the bits are stored in the super block). There are three sets related to backward compatibility:

- ? incompat: not backward compatible, mount on older kernel will fail.
- ? compat_ro: backward compatible for read-only mount.
- ? compat: backward compatible with read-write support, only marked as as individual feature.

??		
? Field	? Description	?
??		
? ioctl fd	? file descriptor of the subvolume ?	
?	? to examine	?
??		
? ioctl args ? struct btrfs_ioctl_feature_flags ?		
??		

If a bit is set then there may be some data structures on the filesystem of the related feature, but not necessarily.

Some of the features are turned on automatically when used, e.g. compression or when a balance filter converts to yet unused block group profile. In some cases the feature can be turned on or off by `btrfstune(8)` <>.

BTRFS_IOC_SET_FEATURES

Set a feature bit on the filesystem if possible. Some features may require extensive changes, new data structures or conversion (like free-space-tree). Bits representing possible existence of data structures related to the feature can be set without actually creating anything, e.g. ZSTD compressed extents.

??		
? Field	? Description	?
??		
? ioctl fd	? file descriptor of the subvolume ?	
?	? to examine	?
??		
? ioctl args ? struct btrfs_ioctl_feature_flags ?		
??		

BTRFS_IOC_GET_SUPPORTED_FEATURES

- Get feature sets supported by the kernel module, in three groups:
- ? supported: (index 0) all supported compat/compat_ro/incompat features
 - ? safe to set: (index 1) features that can be enabled on a mounted filesystem

? safe to clear: (index 2) features that can be disabled on a mounted filesystem

The features are also listed in /sys/fs/btrfs/features.

```

????????????????????????????????????????????????????????
? Field      ? Description              ?
????????????????????????????????????????????????????????
? ioctl fd   ? file descriptor of the subvolume ?
?           ? to examine                ?
????????????????????????????????????????????????????????
? ioctl args ? struct          btrfs_ioctl_fea? ?
?           ? ture_flags[3]      ?
????????????????????????????????????????????????????????

```

BTRFS_IOC_GET_SUBVOL_INFO

Get information about a subvolume.

```

????????????????????????????????????????????????????????
? Field      ? Description              ?
????????????????????????????????????????????????????????
? ioctl fd   ? file descriptor of the subvolume ?
?           ? to examine                ?
????????????????????????????????????????????????????????
? ioctl args ? struct          btrfs_ioctl_get_sub? ?
?           ? vol_info_args      ?
????????????????????????????????????????????????????????

```

BTRFS_IOC_SNAP_DESTROY_V2

Destroy a subvolume, which may or may not be a snapshot.

```

????????????????????????????????????????????????????????
? Field      ? Description              ?
????????????????????????????????????????????????????????
? ioctl fd   ? if flags does not include ?
?           ? BTRFS_SUBVOL_SPEC_BY_ID, or if ?
?           ? executing in a non-root user ?
?           ? namespace, file descriptor of the ?
?           ? parent directory containing the ?
?           ? subvolume to delete; otherwise, ?

```


? file descriptor of any directory ?
 ? on the same filesystem as the ?
 ? subvolume to delete, but not ?
 ? within the same subvolume ?
 ???
 ? ioctl args ? struct btrfs_ioctl_vol_args_v2 ?
 ???
 ? args.fd ? ignored ?
 ???
 ? args.transid ? ignored ?
 ???
 ? args.flags ? 0 if the name field identifies ?
 ? the subvolume by name in the ?
 ? specified directory, or ?
 ? BTRFS_SUBVOL_SPEC_BY_ID if the ?
 ? subvolid field specifies the ID ?
 ? of the subvolume ?
 ???
 ? args.name ? only if flags does not contain ?
 ? BTRFS_SUBVOL_SPEC_BY_ID, the name ?
 ? (within the directory identified ?
 ? by fd) of the subvolume to delete ?
 ???
 ? args.subvolid ? only if flags contains BTRFS_SUB ?
 ? VOL_SPEC_BY_ID, the subvolume ID ?
 ? of the subvolume to delete ?
 ???

BTRFS_IOC_SUBVOL_SYNC_WAIT

(since: 6.13) Wait until a deleted subvolume is cleaned or query the state.

There are several modes of operation, where the most common ones are to wait on a specific subvolume or all currently queued for cleaning. This is utilized e.g. in backup applications that delete subvolumes and wait until they're cleaned to check for remaining space.

The other modes are for flexibility, e.g. for monitoring or checkpoints in the queue of

deleted subvolumes, again without the need to use SEARCH_TREE.

Notes:

? waiting is interruptible, the timeout is set to 1 second and is not configurable

? repeated calls to the ioctl see a different state, so this is inherently racy when using

e.g. the count or peek next/last

Use cases (definition of constants):

? a subvolume A was deleted, wait for cleaning (WAIT_FOR_ONE)

? a bunch of subvolumes were deleted, wait for all (WAIT_FOR_QUEUED or PEEK_LAST +
WAIT_FOR_ONE)

? count how many are queued (not blocking), for monitoring purposes

? report progress (PEEK_NEXT), may miss some if cleaning is quick

? own waiting in user space (PEEK_LAST until it's 0)

??

? Field ? Description ?

??

? ioctl fd ? file descriptor of any file or ?

? ? directory in the filesystem ?

??

? ioctl args ? struct btrfs_ioctl_subvol_wait ?

??

? args.subvolid ? Depending on the mode, the nu? ?

? ? meric id of subvolume to wait ?

? ? for, or the one queried by PEEK ?

? ? modes ?

??

? args.mode ? mode of operation described above ?

??

? args.count ? if mode is set to COUNT the num? ?

? ? ber of subvolumes queued for ?

? ? cleaning ?

??

AVAILABILITY

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

.readthedocs.io>.

SEE ALSO

ioctl(2) <<https://man7.org/linux/man-pages/man2/ioctl.2.html>>

6.17.1

Jan 22, 2026

BTRFS-IOCTL(2)