

**NAME**

`ioctl_fideduperange` – share some the data of one file with another file

**SYNOPSIS**

```
#include <sys/ioctl.h>
```

```
#include <linux/fs.h>
```

```
int ioctl(int src_fd, FIDEDUPERANGE, struct file_dedupe_range *arg);
```

**DESCRIPTION**

If a filesystem supports files sharing physical storage between multiple files, this `ioctl(2)` operation can be used to make some of the data in the `src_fd` file appear in the `dest_fd` file by sharing the underlying storage if the file data is identical ("deduplication"). Both files must reside within the same filesystem. This reduces storage consumption by allowing the filesystem to store one shared copy of the data. If a file write should occur to a shared region, the filesystem must ensure that the changes remain private to the file being written. This behavior is commonly referred to as "copy on write".

This `ioctl` performs the "compare and share if identical" operation on up to `src_length` bytes from file descriptor `src_fd` at offset `src_offset`. This information is conveyed in a structure of the following form:

```
struct file_dedupe_range {
    __u64 src_offset;
    __u64 src_length;
    __u16 dest_count;
    __u16 reserved1;
    __u32 reserved2;
    struct file_dedupe_range_info info[0];
};
```

Deduplication is atomic with regards to concurrent writes, so no locks need to be taken to obtain a consistent deduplicated copy.

The fields `reserved1` and `reserved2` must be zero.

Destinations for the deduplication operation are conveyed in the array at the end of the structure. The number of destinations is given in `dest_count`, and the destination information is conveyed in the following form:

```
struct file_dedupe_range_info {
    __s64 dest_fd;
    __u64 dest_offset;
    __u64 bytes_deduped;
    __s32 status;
    __u32 reserved;
};
```

Each deduplication operation targets `src_length` bytes in file descriptor `dest_fd` at offset `dest_offset`. The field `reserved` must be zero. During the call, `src_fd` must be open for reading and `dest_fd` must be open for writing. The combined size of the struct `file_dedupe_range` and the struct `file_dedupe_range_info` array must not exceed the system page size. The maximum size of `src_length` is filesystem dependent and is typically 16 MiB. This limit will be enforced silently by the filesystem. By convention, the storage used by `src_fd` is mapped into `dest_fd` and the previous contents in `dest_fd` are freed.

Upon successful completion of this `ioctl`, the number of bytes successfully deduplicated is returned in `bytes_deduped` and a status code for the deduplication operation is returned in `status`. If even a single byte in the range does not match, the deduplication request will be ignored and `status` set to **FILE\_DEDUPE\_RANGE\_DIFFERS**. The `status` code is set to **FILE\_DEDUPE\_RANGE\_SAME** for success, a negative error code in case of error, or **FILE\_DEDUPE\_RANGE\_DIFFERS** if the data did not match.

## RETURN VALUE

On error, `-1` is returned, and `errno` is set to indicate the error.

## ERRORS

Error codes can be one of, but are not limited to, the following:

### EBADF

`src_fd` is not open for reading; `dest_fd` is not open for writing or is open for append-only writes; or the filesystem which `src_fd` resides on does not support deduplication.

### EINVAL

The filesystem does not support deduplicating the ranges of the given files. This error can also appear if either file descriptor represents a device, FIFO, or socket. Disk filesystems generally require the offset and length arguments to be aligned to the fundamental block size. Neither Btrfs nor XFS support overlapping deduplication ranges in the same file.

### EISDIR

One of the files is a directory and the filesystem does not support shared regions in directories.

### ENOMEM

The kernel was unable to allocate sufficient memory to perform the operation or `dest_count` is so large that the input argument description spans more than a single page of memory.

### EOPNOTSUPP

This can appear if the filesystem does not support deduplicating either file descriptor, or if either file descriptor refers to special inodes.

### EPERM

`dest_fd` is immutable.

### ETXTBSY

One of the files is a swap file. Swap files cannot share storage.

### EXDEV

`dest_fd` and `src_fd` are not on the same mounted filesystem.

## VERSIONS

This ioctl operation first appeared in Linux 4.5. It was previously known as `BTRFS_IOC_FILE_EXTENT_SAME` and was private to Btrfs.

## CONFORMING TO

This API is Linux-specific.

## NOTES

Because a copy-on-write operation requires the allocation of new storage, the `fallocate(2)` operation may unshare shared blocks to guarantee that subsequent writes will not fail because of lack of disk space.

Some filesystems may limit the amount of data that can be deduplicated in a single call.

## SEE ALSO

`ioctl(2)`

## COLOPHON

This page is part of release 5.05 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.