NAME

quotactl − manipulate disk quotas

SYNOPSIS

#include <sys/quota.h>
#include <xfs/xqm.h> /* for XFS quotas */

int quotactl(int *cmd***, const char ****special***, int** *id***, caddr_t** *addr***);**

DESCRIPTION

The quota system can be used to set per-user, per-group, and per-project limits on the amount of disk space used on a filesystem. For each user and/or group, a soft limit and a hard limit can be set for each filesystem. The hard limit can't be exceeded. The soft limit can be exceeded, but warnings will ensue. Moreover, the user can't exceed the soft limit for more than grace period duration (one week by default) at a time; after this, the soft limit counts as a hard limit.

The **quotactl**() call manipulates disk quotas. The *cmd* argument indicates a command to be applied to the user or group ID specified in *id*. To initialize the *cmd* argument, use the *QCMD(subcmd, type)* macro. The *type* value is either **USRQUOTA**, for user quotas, **GRPQUOTA**, for group quotas, or (since Linux 4.1) **PRJQUOTA**, for project quotas. The *subcmd* value is described below.

The *special* argument is a pointer to a null-terminated string containing the pathname of the (mounted) block special device for the filesystem being manipulated.

The *addr* argument is the address of an optional, command-specific, data structure that is copied in or out of the system. The interpretation of *addr* is given with each command below.

The *subcmd* value is one of the following:

Q_QUOTAON

Turn on quotas for a filesystem. The *id* argument is the identification number of the quota format to be used. Currently, there are three supported quota formats:

QFMT_VFS_OLD

The original quota format.

QFMT_VFS_V0

The standard VFS v0 quota format, which can handle 32-bit UIDs and GIDs and quota limits up to 2^42 bytes and 2^32 inodes.

QFMT_VFS_V1

A quota format that can handle 32-bit UIDs and GIDs and quota limits of 2^64 bytes and 2^64 inodes.

The *addr* argument points to the pathname of a file containing the quotas for the filesystem. The quota file must exist; it is normally created with the **quotacheck**(8) program. This operation requires privilege (**CAP_SYS_ADMIN**).

Q_QUOTAOFF

Turn off quotas for a filesystem. The *addr* and *id* arguments are ignored. This operation requires privilege (**CAP_SYS_ADMIN**).

Q_GETQUOTA

Get disk quota limits and current usage for user or group *id*. The *addr* argument is a pointer to a *dqblk* structure defined in *<sys/quota.h>* as follows:

```
/* uint64_t is an unsigned 64-bit integer;
   uint32_t is an unsigned 32-bit integer */

struct dqblk {      /* Definition since Linux 2.4.22 */
    uint64_t dqb_bhardlimit;  /* Absolute limit on disk
                                 quota blocks alloc */
    uint64_t dqb_bsoftlimit;  /* Preferred limit on
```

```
                                    disk quota blocks */
    uint64_t dqb_curspace;    /* Current occupied space
                                    (in bytes) */
    uint64_t dqb_ihardlimit;  /* Maximum number of
                                    allocated inodes */
    uint64_t dqb_isoftlimit;  /* Preferred inode limit */
    uint64_t dqb_curinodes;   /* Current number of
                                    allocated inodes */
    uint64_t dqb_btime;       /* Time limit for excessive
                                    disk use */
    uint64_t dqb_itime;       /* Time limit for excessive
                                    files */
    uint32_t dqb_valid;       /* Bit mask of QIF_*
                                    constants */
};

/* Flags in dqb_valid that indicate which fields in
   dqblk structure are valid. */

#define QIF_BLIMITS   1
#define QIF_SPACE     2
#define QIF_ILIMITS   4
#define QIF_INODES    8
#define QIF_BTIME     16
#define QIF_ITIME     32
#define QIF_LIMITS    (QIF_BLIMITS | QIF_ILIMITS)
#define QIF_USAGE     (QIF_SPACE | QIF_INODES)
#define QIF_TIMES     (QIF_BTIME | QIF_ITIME)
#define QIF_ALL       (QIF_LIMITS | QIF_USAGE | QIF_TIMES)
```

The *dqb_valid* field is a bit mask that is set to indicate the entries in the *dqblk* structure that are valid. Currently, the kernel fills in all entries of the *dqblk* structure and marks them as valid in the *dqb_valid* field. Unprivileged users may retrieve only their own quotas; a privileged user (**CAP_SYS_ADMIN**) can retrieve the quotas of any user.

Q_GETNEXTQUOTA (since Linux 4.6)

This operation is the same as **Q_GETQUOTA**, but it returns quota information for the next ID greater than or equal to *id* that has a quota set.

The *addr* argument is a pointer to a *nextdqblk* structure whose fields are as for the *dqblk*, except for the addition of a *dqb_id* field that is used to return the ID for which quota information is being returned:

```
struct nextdqblk {
    uint64_t dqb_bhardlimit;
    uint64_t dqb_bsoftlimit;
    uint64_t dqb_curspace;
    uint64_t dqb_ihardlimit;
    uint64_t dqb_isoftlimit;
    uint64_t dqb_curinodes;
    uint64_t dqb_btime;
    uint64_t dqb_itime;
    uint32_t dqb_valid;
    uint32_t dqb_id;
};
```

Q_SETQUOTA
Set quota information for user or group *id*, using the information supplied in the *dqblk* structure pointed to by *addr*. The *dqb_valid* field of the *dqblk* structure indicates which entries in the structure have been set by the caller. This operation supersedes the **Q_SETQLIM** and **Q_SETUSE** operations in the previous quota interfaces. This operation requires privilege (**CAP_SYS_ADMIN**).

Q_GETINFO (since Linux 2.4.22)
Get information (like grace times) about quotafile. The *addr* argument should be a pointer to a *dqinfo* structure. This structure is defined in *<sys/quota.h>* as follows:

```
/* uint64_t is an unsigned 64-bit integer;
   uint32_t is an unsigned 32-bit integer */

struct dqinfo {         /* Defined since kernel 2.4.22 */
    uint64_t dqi_bgrace;  /* Time before block soft limit
                             becomes hard limit */
    uint64_t dqi_igrace;  /* Time before inode soft limit
                             becomes hard limit */
    uint32_t dqi_flags;   /* Flags for quotafile
                             (DQF_*) */
    uint32_t dqi_valid;
};

/* Bits for dqi_flags */

/* Quota format QFMT_VFS_OLD */

#define DQF_ROOT_SQUASH (1 << 0) /* Root squash enabled */
              /* Before Linux v4.0, this had been defined
                 privately as V1_DQF_RSQUASH */

/* Quota format QFMT_VFS_V0 / QFMT_VFS_V1 */

#define DQF_SYS_FILE    (1 << 16)   /* Quota stored in
                                       a system file */

/* Flags in dqi_valid that indicate which fields in
   dqinfo structure are valid. */

#define IIF_BGRACE  1
#define IIF_IGRACE  2
#define IIF_FLAGS   4
#define IIF_ALL     (IIF_BGRACE | IIF_IGRACE | IIF_FLAGS)
```

The *dqi_valid* field in the *dqinfo* structure indicates the entries in the structure that are valid. Currently, the kernel fills in all entries of the *dqinfo* structure and marks them all as valid in the *dqi_valid* field. The *id* argument is ignored.

Q_SETINFO (since Linux 2.4.22)
Set information about quotafile. The *addr* argument should be a pointer to a *dqinfo* structure. The *dqi_valid* field of the *dqinfo* structure indicates the entries in the structure that have been set by the caller. This operation supersedes the **Q_SETGRACE** and **Q_SETFLAGS** operations in the previous quota interfaces. The *id* argument is ignored. This operation requires privilege (**CAP_SYS_ADMIN**).

Q_GETFMT (since Linux 2.4.22)
Get quota format used on the specified filesystem. The *addr* argument should be a pointer to a 4-byte buffer where the format number will be stored.

Q_SYNC
Update the on-disk copy of quota usages for a filesystem. If *special* is NULL, then all filesystems with active quotas are sync'ed. The *addr* and *id* arguments are ignored.

Q_GETSTATS (supported up to Linux 2.4.21)
Get statistics and other generic information about the quota subsystem. The *addr* argument should be a pointer to a *dqstats* structure in which data should be stored. This structure is defined in *<sys/quota.h>*. The *special* and *id* arguments are ignored.

This operation is obsolete and was removed in Linux 2.4.22. Files in */proc/sys/fs/quota/* carry the information instead.

For XFS filesystems making use of the XFS Quota Manager (XQM), the above commands are bypassed and the following commands are used:

Q_XQUOTAON
Turn on quotas for an XFS filesystem. XFS provides the ability to turn on/off quota limit enforcement with quota accounting. Therefore, XFS expects *addr* to be a pointer to an *unsigned int* that contains a combination of the following flags (defined in *<xfs/xqm.h>*):

```
#define XFS_QUOTA_UDQ_ACCT (1<<0) /* User quota
                                     accounting */
#define XFS_QUOTA_UDQ_ENFD (1<<1) /* User quota limits
                                     enforcement */
#define XFS_QUOTA_GDQ_ACCT (1<<2) /* Group quota
                                     accounting */
#define XFS_QUOTA_GDQ_ENFD (1<<3) /* Group quota limits
                                     enforcement */
#define XFS_QUOTA_PDQ_ACCT (1<<4) /* Project quota
                                     accounting */
#define XFS_QUOTA_PDQ_ENFD (1<<5) /* Project quota limits
                                     enforcement */
```

This operation requires privilege (**CAP_SYS_ADMIN**). The *id* argument is ignored.

Q_XQUOTAOFF
Turn off quotas for an XFS filesystem. As with **Q_QUOTAON**, XFS filesystems expect a pointer to an *unsigned int* that specifies whether quota accounting and/or limit enforcement need to be turned off (using the same flags as for **Q_XQUOTAON** subcommand). This operation requires privilege (**CAP_SYS_ADMIN**). The *id* argument is ignored.

Q_XGETQUOTA
Get disk quota limits and current usage for user *id*. The *addr* argument is a pointer to an *fs_disk_quota* structure, which is defined in *<xfs/xqm.h>* as follows:

```
/* All the blk units are in BBs (Basic Blocks) of
   512 bytes. */

#define FS_DQUOT_VERSION  1  /* fs_disk_quota.d_version */

#define XFS_USER_QUOTA    (1<<0)  /* User quota type */
#define XFS_PROJ_QUOTA    (1<<1)  /* Project quota type */
#define XFS_GROUP_QUOTA   (1<<2)  /* Group quota type */

struct fs_disk_quota {
    int8_t   d_version;   /* Version of this structure */
```

```
    int8_t   d_flags;     /* XFS_{USER,PROJ,GROUP}_QUOTA */
    uint16_t d_fieldmask; /* Field specifier */
    uint32_t d_id;        /* User, project, or group ID */
    uint64_t d_blk_hardlimit; /* Absolute limit on
                                 disk blocks */
    uint64_t d_blk_softlimit; /* Preferred limit on
                                 disk blocks */
    uint64_t d_ino_hardlimit; /* Maximum # allocated
                                 inodes */
    uint64_t d_ino_softlimit; /* Preferred inode limit */
    uint64_t d_bcount;    /* # disk blocks owned by
                             the user */
    uint64_t d_icount;    /* # inodes owned by the user */
    int32_t  d_itimer;    /* Zero if within inode limits */
                          /* If not, we refuse service */
    int32_t  d_btimer;    /* Similar to above; for
                             disk blocks */
    uint16_t d_iwarns;    /* # warnings issued with
                             respect to # of inodes */
    uint16_t d_bwarns;    /* # warnings issued with
                             respect to disk blocks */
    int32_t  d_padding2;  /* Padding - for future use */
    uint64_t d_rtb_hardlimit; /* Absolute limit on realtime
                                 (RT) disk blocks */
    uint64_t d_rtb_softlimit; /* Preferred limit on RT
                                 disk blocks */
    uint64_t d_rtbcount;  /* # realtime blocks owned */
    int32_t  d_rtbtimer;  /* Similar to above; for RT
                             disk blocks */
    uint16_t d_rtbwarns;  /* # warnings issued with
                             respect to RT disk blocks */
    int16_t  d_padding3;  /* Padding - for future use */
    char     d_padding4[8];   /* Yet more padding */
};
```

Unprivileged users may retrieve only their own quotas; a privileged user (**CAP_SYS_ADMIN**) may retrieve the quotas of any user.

Q_XGETNEXTQUOTA (since Linux 4.6)
This operation is the same as **Q_XGETQUOTA**, but it returns (in the *fs_disk_quota* structure pointed by *addr*) quota information for the next ID greater than or equal to *id* that has a quota set. Note that since *fs_disk_quota* already has *q_id* field, no separate structure type is needed (in contrast with **Q_GETQUOTA** and **Q_GETNEXTQUOTA** commands)

Q_XSETQLIM
Set disk quota limits for user *id*. The *addr* argument is a pointer to an *fs_disk_quota* structure. This operation requires privilege (**CAP_SYS_ADMIN**).

Q_XGETQSTAT
Returns XFS filesystem-specific quota information in the *fs_quota_stat* structure pointed by *addr*. This is useful for finding out how much space is used to store quota information, and also to get the quota on/off status of a given local XFS filesystem. The *fs_quota_stat* structure itself is defined as follows:

```
#define FS_QSTAT_VERSION 1  /* fs_quota_stat.qs_version */

struct fs_qfilestat {
```

```
    uint64_t qfs_ino;       /* Inode number */
    uint64_t qfs_nblks;     /* Number of BBs
                               512-byte-blocks */
    uint32_t qfs_nextents;  /* Number of extents */
};

struct fs_quota_stat {
    int8_t   qs_version; /* Version number for
                            future changes */
    uint16_t qs_flags; /* XFS_QUOTA_{U,P,G}DQ_{ACCT,ENFD} */
    int8_t   qs_pad;   /* Unused */
    struct fs_qfilestat qs_uquota;  /* User quota storage
                                       information */
    struct fs_qfilestat qs_gquota;  /* Group quota storage
                                       information */
    uint32_t qs_incoredqs;   /* Number of dquots in core */
    int32_t  qs_btimelimit;  /* Limit for blocks timer */
    int32_t  qs_itimelimit;  /* Limit for inodes timer */
    int32_t  qs_rtbtimelimit;/* Limit for RT
                                blocks timer */
    uint16_t qs_bwarnlimit;  /* Limit for # of warnings */
    uint16_t qs_iwarnlimit;  /* Limit for # of warnings */
};
```

The *id* argument is ignored.

Q_XGETQSTATV

Returns XFS filesystem-specific quota information in the *fs_quota_statv* pointed to by *addr*. This version of the command uses a structure with proper versioning support, along with appropriate layout (all fields are naturally aligned) and padding to avoiding special compat handling; it also provides the ability to get statistics regarding the project quota file. The *fs_quota_statv* structure itself is defined as follows:

```
#define FS_QSTATV_VERSION1 1 /* fs_quota_statv.qs_version */

struct fs_qfilestatv {
    uint64_t qfs_ino;       /* Inode number */
    uint64_t qfs_nblks;     /* Number of BBs
                               512-byte-blocks */
    uint32_t qfs_nextents;  /* Number of extents */
    uint32_t qfs_pad;       /* Pad for 8-byte alignment */
};

struct fs_quota_statv {
    int8_t   qs_version;    /* Version for future
                               changes */
    uint8_t  qs_pad1;       /* Pad for 16-bit alignment */
    uint16_t qs_flags;      /* XFS_QUOTA_.* flags */
    uint32_t qs_incoredqs;  /* Number of dquots incore */
    struct fs_qfilestatv qs_uquota;  /* User quota
                                        information */
    struct fs_qfilestatv qs_gquota;  /* Group quota
                                        information */
    struct fs_qfilestatv qs_pquota;  /* Project quota
                                        information */
    int32_t  qs_btimelimit;   /* Limit for blocks timer */
```

```
    int32_t  qs_itimelimit;   /* Limit for inodes timer */
    int32_t  qs_rtbtimelimit; /* Limit for RT blocks
                                 timer */
    uint16_t qs_bwarnlimit;   /* Limit for # of warnings */
    uint16_t qs_iwarnlimit;   /* Limit for # of warnings */
    uint64_t qs_pad2[8];      /* For future proofing */
};
```

The *qs_version* field of the structure should be filled with the version of the structure supported by the callee (for now, only *FS_QSTAT_VERSION1* is supported). The kernel will fill the structure in accordance with version provided. The *id* argument is ignored.

Q_XQUOTARM
Free the disk space taken by disk quotas. The *addr* argument should be a pointer to an *unsigned int* value containing flags (the same as in *d_flags* field of *fs_disk_quota* structure) which identify what types of quota should be removed (note that the quota type passed in the *cmd* argument is ignored, but should remain valid in order to pass preliminary quotactl syscall handler checks).

Quotas must have already been turned off. The *id* argument is ignored.

Q_XQUOTASYNC (since Linux 2.6.15; no-op since Linux 3.4)
This command was an XFS quota equivalent to *Q_SYNC*, but it is no-op since Linux 3.4, as **sync**(1) writes quota information to disk now (in addition to the other filesystem metadata that it writes out). The *special*, *id* and *addr* arguments are ignored.

RETURN VALUE

On success, **quotactl**() returns 0; on error −1 is returned, and *errno* is set to indicate the error.

ERRORS

EACCES
cmd is **Q_QUOTAON**, and the quota file pointed to by *addr* exists, but is not a regular file or is not on the filesystem pointed to by *special*.

EBUSY
cmd is **Q_QUOTAON**, but another **Q_QUOTAON** had already been performed.

EFAULT
addr or *special* is invalid.

EINVAL
cmd or *type* is invalid.

EINVAL
cmd is **Q_QUOTAON**, but the specified quota file is corrupted.

ENOENT
The file specified by *special* or *addr* does not exist.

ENOSYS
The kernel has not been compiled with the **CONFIG_QUOTA** option.

ENOTBLK
special is not a block device.

EPERM
The caller lacked the required privilege (**CAP_SYS_ADMIN**) for the specified operation.

ERANGE
cmd is **Q_SETQUOTA**, but the specified limits are out of the range allowed by the quota format.

ESRCH
No disk quota is found for the indicated user. Quotas have not been turned on for this filesystem.

ESRCH
cmd is **Q_QUOTAON**, but the specified quota format was not found.

ESRCH
cmd is **Q_GETNEXTQUOTA** or **Q_XGETNEXTQUOTA**, but there is no ID greater than or equal to *id* that has an active quota.

NOTES

Instead of *<xfs/xqm.h>* one can use *<linux/dqblk_xfs.h>*, taking into account that there are several naming discrepancies:

- Quota enabling flags (of format **XFS_QUOTA_[UGP]DQ_{ACCT,ENFD}**) are defined without a leading "X", as **FS_QUOTA_[UGP]DQ_{ACCT,ENFD}**.
- The same is true for **XFS_{USER,GROUP,PROJ}_QUOTA** quota type flags, which are defined as **FS_{USER,GROUP,PROJ}_QUOTA**.
- The *dqblk_xfs.h* header file defines its own **XQM_USRQUOTA**, **XQM_GRPQUOTA**, and **XQM_PRJQUOTA** constants for the available quota types, but their values are the same as for constants without the **XQM_** prefix.

SEE ALSO

quota(1), **getrlimit**(2), **quotacheck**(8), **quotaon**(8)

COLOPHON

This page is part of release 5.02 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/.