

Bahir Dar University

Institute of Technology

**Faculty of Computing**

**Department of Software Engineering**

**Opereating systems and system programming assignment II**

**Title: quotactrl()**

By Fikiremariyam.b

1311608

Course: os &sp

Submitted to: Wodimu B.

Submission Date: July 24 , 2022

**Overview of the document**

* [Why the system call](#_why_the_system)
* [Detail description of parameters of the command](#_The_parameters)
* [The sub commands of the main command](#_sub_commands)
* [Step by step implantation and flag usage](#_Quta_ctrl_detail)
* [Syntax using the qctrl command](#_Implementation_example)

# why the system call ?

The Quota ctl command:

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 file system. For each user and/or group, a soft limit and a hard limit can beset for each file system.

In file system quota. Instead of allowing every user to use all of the available space on a filesystem (or drive, to be less technical) each user is confined to limits – so called quotas. These quotas determine how much space an individual user or group may use.

Linux libc exposes support for controlling filesystem quotas via its quotactl function. musl-libc shows that quotactl can be a thin wrapper around the underlying syscall of the same name:

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

{

return syscall(SYS\_quotactl, cmd, special, id, addr);

}

This function, along with a few C preprocessor macros which are defined in musl‘s include/sys/quota.h, provides full filesystem quota support

# The parameters

**int quotactl(int** *cmd***, const char \****special***, int** *id***, caddr\_t** *addr***);**

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- speific, data

structure that is copied in or out of the system.

# sub commands

The *subcmd* value is one of the following operations: this are the commands we use inside a command in order to do specific task .they return specific value.

1. **Q\_QUOTAON**

Turn on quotas for a filesystem.i.e enable enforcement of quota limits

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](https://man7.org/linux/man-pages/man8/quotacheck.8.html) program

2 **Q\_QUOTAOFF**

Turn off quotas for a filesystem. The *addr* and *id*

arguments are ignored. This operation requires privilege

**3 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>*

**4 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.

**5 Q\_SETQUOTA**

Set quota information for user or group *id*, using the

information supplied in the *dqblk* structure pointed to by

*addr*.

**6 Q\_XQUOTAON**

Turn on quotas for an XFS filesystem. XFS provides the

ability to turn on/off quota limit enforcement with quota

accounting.

**7 Q\_XQUOTAOFF**

Turn off quotas for an XFS filesystem. As with 8 8 **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** operation).

**9 Q\_XGETQUOTA**

Get disk quota limits and current usage for user *id*.

10 Q **\_XGETQSTATV**

Returns XFS filesystem-specific quota information in the

*fs\_quota\_statv* pointed to by *addr*.

# 

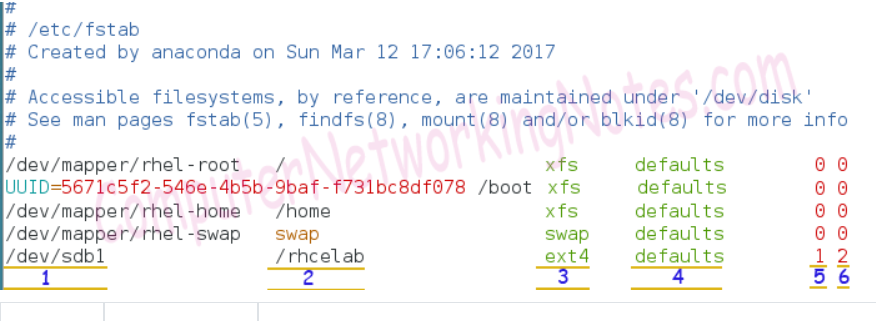
# Quta ctrl detail flags and their implementation

In other commands flags of the commands are used to describe the detail application and data of the certain command here the application of flag are executed by –g-p-–root and –mount flags which we use them in dffrent steps

## **How to configure the disk quota**

1. Enable quota
2. Remount file system
3. Create quota files
4. Configure quota policy

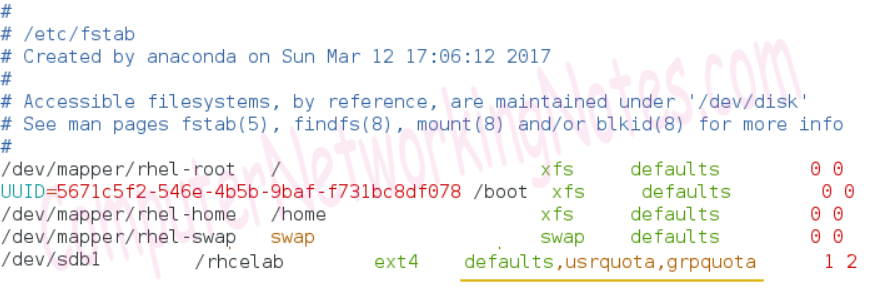
linux uses */etc/fstab* configuration file to mount all partitions in file system at boot time. This file contains all necessary information about the partition such as partition location at disk, mount point, attributes and other control options which are required to mount a partition. Each entry in this file has six fields.



| Number | Filed | Description |
| --- | --- | --- |
| 1 | What to mount | Device which we want to mount.  We can use device name, UUID and label in this filed to represent the  device. |
| 2 | Where to mount | The directory in main Linux File System where we want to mount the  device. |
| 3 | File system | File system type of device. |
| 4 | Options | Mount options which control the mount process. To enable user  quota add  **usrquota** option and to enable group quota add **grpquota** option. |
| 5 | Dump support | To enable the dump on this device use 1. Use 0 to disable the dump. |
| 6 | Automatic check | Whether this device should be checked while mounting or not.  To disable use 0, to enable use 1 (for root partition) or 2 (for all  partitions except root partition) |
|  |  |  |

In order to enable user quota, we have to add *usrquota* option in fourth field. Just like it, to enable group quota, we have to add *grpquota* option in fourth field. Let’s enable both quotas for partition /dev/sdb1.

Following figure illustrates updated */etc/fstab* file



Remounting file system

#mount –o remount [partition]

### **Creating quota files**

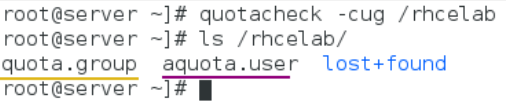
#quotacheck

**quotacheck** :- This command is used to check the quota implementation in partition.

**c** :- This option is used to create the quota files in specified partition.

**u** :- This option is used to check the user quota.

**g** :- This option is used to check the group quota



### **Configuring quota policies**

To configure quota policies, we have to define three values; soft limit, hard limit and grace period.

**Soft limit**: - This limit is flexible. User or group is allowed to cross this limit temporary.

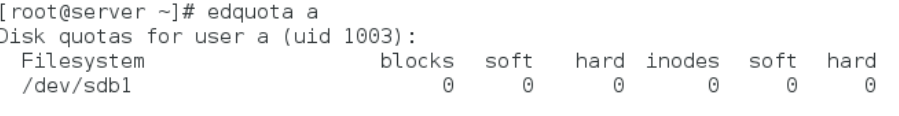
**Hard limit**: - This is fixed limit. User or group is not allowed to cross this limit.

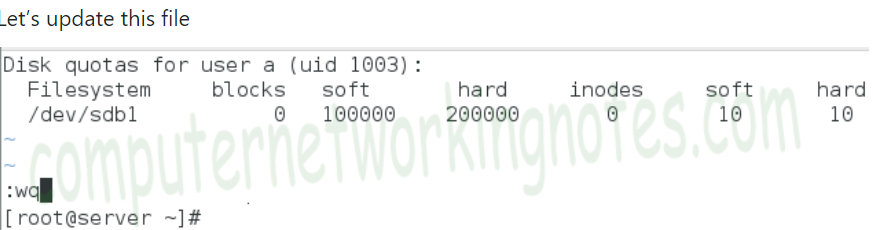
**Grace period**: - This is the time period in which user or group is allowed to use additional space beyond the soft limit.

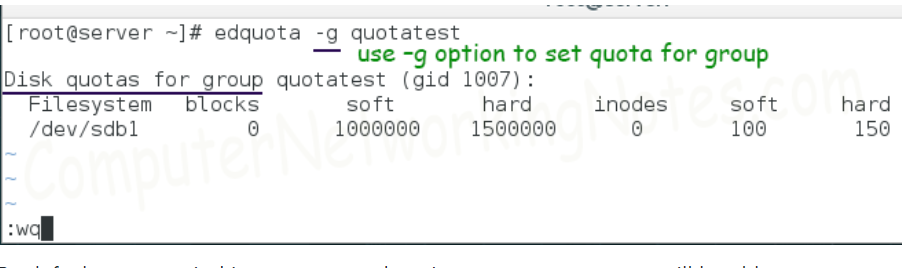
configure quota

To configure quota for user **a** use following command

#edquota a







# Implementation example

Another interesting detail we need to have a closer look at is how the cmd argument to the quotactl call is built. The man page references the QCMD(subcmd, type) macro, which is defined in sys/quota.h, taking a “subcommand” and a “quota type”, and applies a bitwise shift operation, followed by masking the type with an “AND” operation.

Here is a function that access the the information of the punter of the quota settled

#define QCMD(cmd, type) (((cmd) << SUBCMDSHIFT) | ((type) & SUBCMDMASK))

func GetQuota(typ int, special string, id int) (result \*Dqblk, err error) {

result = &Dqblk{}

if err = quotactl(qCmd(qGetQuota, typ), special, id, unsafe.Pointer(result)); err != nil {

result = nil

}

return

}

Reffrence

: <https://www.computernetworkingnotes.com/linux-tutorials/how-to-manage-disk-quota-in-linux-step-by-step.html>

<https://man7.org/linux/man-pages/man2/quotactl.2.html>

https://anexia.com/blog/en/filesystem-quota-management-in-go/