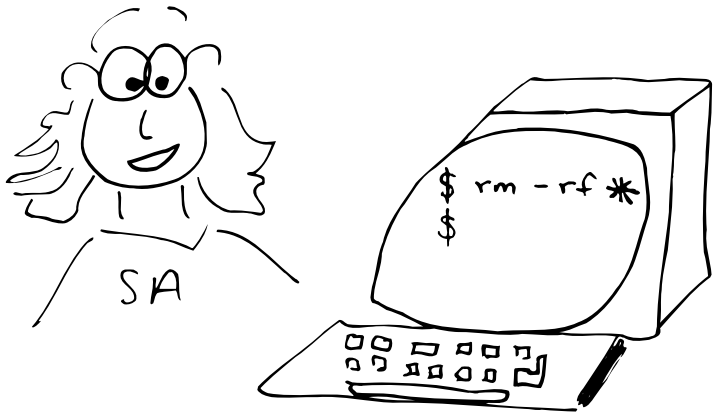
26 Disk Space Management



26.1 Introduction

It has been said that the only thing all UNIX systems have in common is the login message asking users to clean up their files and use less disk space. No matter how much space you have, it isn't enough; as soon as a disk is added, files magically appear to fill it up.

Both users and the system itself are potential sources of disk bloat. Chapter 12, *Syslog and Log Files*, discusses various sources of logging information and the techniques used to manage them. This chapter focuses on space problems caused by users and the technical and psychological weapons you can deploy against them.

If you do decide to add a disk, refer to Chapter 9 for help.

Even if you have the option of adding more disk storage to your system, it's a good idea to follow this chapter's suggestions. Disks are cheap, but administrative effort is not. Disks have to be dumped, maintained, cross-mounted, and monitored; the fewer you need, the better.

26.2 Dealing with Disk Hogs

In the absence of external pressure, there is essentially no reason for a user to ever delete anything. It takes time and effort to clean up unwanted files, and there's always the risk that something thrown away might be wanted again in the future. Even when users have good intentions, it often takes a nudge from the system administrator to goad them into action.

On a PC, disk space eventually runs out and the machine's primary user must clean up to get the system working again. But on a UNIX machine, many users can share a disk. When space gets low, users sometimes try to ignore the problem as long as they can in the hope that someone else will "break" first. It's often hard to convince users that they should remove any of their precious files until the disk is actually full or overflowing. Some users keep large junk files around just so that they'll have something to delete when the disk fills up and they can no longer get any work done.

It does not work to send mail to all users asking them to clean up their files or to post a message about the problem in **/etc/motd**. These methods don't assign responsibility to specific people. To get action, you have to find out who the disk hogs are and let them know that *you* know they are the source of the problem.

***spacegripe** is included on the CD-ROM.*

You can do this automatically with a script that calculates disk usage for each user, identifies those whose consumption is above a certain threshold, and sends polite mail requesting that they clean up their files. We call our version of this script **spacegripe**. Since **spacegripe** needs to forage in users' home directories, it must be run as root. You can set the threshold at which mail is sent by replacing the number 10,000 with the maximum number of disk blocks someone can have without being pestered.

spacegripe is quite polite and precise, but alas, it is generally ignored by our user community. It's most effective the first time a user receives a message; after that, the novelty wears off and subsequent messages are often deleted without being read. Since the mail does not come from a real person, it's perceived as being only slightly more personal than a broadcast message.

No one likes to be labeled as one of the top ten disk hogs, especially if disk space is tight enough that other users are having trouble getting their work done. We have found that publishing such a list is by far the most effective way of "persuading" users to clean up. Whenever a list of disk hogs is posted in **/etc/motd**, the disk space situation miraculously improves.[1]

If some users do not reduce their disk usage even after being publicly denounced, you will have to deal with them on a person-by-person basis. Be gentle; a friendly message from an administrator has ten times the impact of an automated reminder.

Another option for automation is to compress files that are larger than a certain threshold and that have not been accessed recently, say in

1. At sites where every user has a workstation, people tend to stay logged in all the time and therefore never see the contents of **/etc/motd**. Public email is a good substitute.

thirty days. This is an invasive tactic and it is not 100% safe, since users' files must be modified. However, it does free up a lot of disk space and is worth considering in extreme cases.

See page 621 for more information about compression.

A **perl** script called **compressfs** is included on the CD-ROM; it performs the compression chores and then sends email to each user whose files were compressed to explain what has happened.

When you ask users to clean up, you will get better results if you provide an easy way for them to store files off-line. A tape drive in a public area allows users to archive infrequently-used files with minimal help from you. In a semi-public setting such as a university, you might want to consider selling tapes. DAT and QIC tapes can be hard to find, and it takes some familiarity with the media to know what to buy. At minimum, attach information to the tape drive that describes what kind of media to buy, where to find it, and how much it costs.

26.3 HOG DETECTION

Information about disk usage can be obtained with the **quot** command, which shows each user's total number of files and disk blocks on each filesystem. For example, **quot -f /dev/sd4c** produces

```
blocks   files   user
-------------------------
/dev/sd4c (/home/anchor):
112180   2501   markey
66340    3254   drew
63258    1267   weinberj
53874    5918   christos
45192    9761   jules
...
```

The **quot** command is *not* related to the quota system discussed later in this chapter. **du** summarizes the disk usage within a directory hierarchy. For example, **du -s /home/anchor/*** yields

```
blocks  user
--------------
112325  markey
66332   drew
63258   weinberj
53874   christos
47311   jules
...
```

The numbers reported by these commands are in "disk blocks." Unfortunately, folks and filesystems can't seem to agree on how big a block is. Table 26.1 shows the block sizes for various operating systems, in bytes. Block size is actually a parameter of each filesystem, but many com-

mands don't take this fact into consideration. Files with holes[2] should not be expanded when measuring file sizes, but on some systems, with some commands, they are. Database files created by **dbm** always contain holes and are usually only 25% of their apparent size.

Table 26.1 Block sizes used by various commands

System	du	df	quot
Solaris	512[a]	512[a]	1024
HP-UX	512	512	1024
IRIX	512[a]	512[a]	1024
SunOS	1024	1024	1024
OSF/1	512[a]	512[a]	1024
BSDI	512[b]	1024	–

a. You can get 1K blocks with the **-k** option.

b. Uses environment variable BLOCKSIZE, if defined.

The HP-UX manual page claims that **quot** uses 2,048-byte blocks, which is true for **quot -h**, but not true for the **-f**, **-c**, and **-v** options. HP-UX provides the Berkeley version of **df** under the name **bdf**.

quot counts all files belonging to a user; **du** counts all files in a particular directory. Users can own files outside their home directories, and there can be files in users' home directories that don't belong to them. Thus, there may be a discrepancy between the numbers reported by **du** and **quot**. Holes in files and the counting algorithm for symbolic links also influence the reported sizes.

26.4 DATA COMPRESSION

Most UNIX systems provide at least one set of utilities for data compression and expansion. These utilities usually include a compression program, an expansion program, and a program that dynamically expands for viewing. Some common program sets are the **compress** family, the **gzip** family, and the **pack** family.

***gzip** is a GNU thing. It's included on the CD-ROM.*

The best compression ratios are achieved with **gzip**, but it is fairly slow and not all systems provide it. There are some compatibility problems with early versions of the command, so if you use **gzip** it is wise to standardize on the most recent version.

2. A file that is created by a program that writes a byte, seeks out a megabyte, and then writes another byte is called a file with a hole in it. Should it occupy two bytes on the disk or a million and two? Files with holes are usually stored with the holes compacted; they are sometimes expanded by programs that either measure their size (**du** under ATT) or archive them (**tar** or **cpio**).

compress is peppier than **gzip** and is universally available; its compression is pretty good, too. **pack** is obsolete and should not be used if you have a choice. It is even faster than **compress**, but it provides relatively poor compression. Table 26.2 compares the performance of the **compress**, **gzip**, and **pack** commands.

Table 26.2 Comparison of compress, gzip, and pack

	compress		gzip		pack	
Input	Saved[a]	Time	Saved[a]	Time	Saved[a]	Time
2.1MB English text	57.8%	16.3 s	61.4%	50.0 s	38.9%	8.8 s
1.8MB Binary file	50.0%	14.2 s	61.9%	43.2 s	25.1%	8.1 s
3.3MB C code	60.4%	24.1 s	74.0%	51.4 s	35.5%	14.3 s
2.6MB Encrypted	none		none		none	

a. Percentage of original size removed. Bigger numbers indicate better compression.

Encrypted data does not compress.[3] Superficially, it appears to be random data and thus fools the compression algorithms, which look for patterns. There are other kinds of data that do not compress or that compress poorly; for example, DNA sequencing information. Compressed files generally cannot be compressed again.

Large files that are only accessed occasionally are good targets for compression. When deciding whether to compress a file, you must decide whether the savings in disk space warrant the CPU time and the hassle that it takes to compress and expand the file.

26.5 SKULKER SCRIPTS

skulker is the name usually given to a script that goes around the disk, controlling the size of system logs, removing abandoned junk files, and checking for security breaches. **skulker** scripts are usually run by **cron** either daily or weekly.

The junk files that **skulker** should remove vary from system to system. Editor checkpoint and backup files, core files, and certain by-products of compilation are generally safe to remove, but there is always a chance that someone will unknowingly name an important file to match one of **skulker**'s specifications and have it deleted. Your site's deletion policies should be documented in a public place so that users will not be surprised when their files disappear.

Cleaning the Filesystem on page 176 gives examples of commands that might be used in a **skulker** script. Many of the security-related com-

3. Actually, it gets bigger when you try to compress it.

mands described starting on page 549 are also good candidates for inclusion in a **skulker**.

26.6 TUNEFS: SET FILESYSTEM PARAMETERS

The **tunefs** command is used to alter the layout policies on a particular filesystem. **tunefs** doesn't change the filesystem's contents; it simply modifies the way in which future write operations will be handled.

The following filesystem parameters can be adjusted with **tunefs**:

- The rotational delay between groups of blocks in a file
- The maximum number of blocks in a single transfer
- The number of blocks a file may claim from a cylinder group
- The amount of disk space to reserve as overhead

Theoretically, you can manipulate these parameters (and others specified when the filesystem is constructed) to achieve better filesystem performance. Unfortunately, most modern hard drives are geometrically more complex than UNIX expects, and the kernel's optimizations are often less than perfectly effective. Some settings make a difference and some don't; most sites don't bother to fiddle with the defaults.

The overhead or "reserve" setting is still a useful tool, however. If this parameter is set to a value other than zero, the filesystem conceals a percentage of the available disk space. The default reserve is 10%, causing writes to fail when the filesystem becomes 90% full. The remaining 10% of space can only be used by root.[4]

This accounting trick allows the filesystem to keep pockets of free space together so that new files can be written out in large chunks, boosting performance. On a full disk with no reserve, throughput can be three times slower than on a disk that's 90% full.

If you're completely out of disk space, can't delete anything, and can't add more storage to the system, you might consider shrinking the filesystem reserve to free up more space. The command

```
tunefs -m pctfree device
```

sets the reserve percentage for the filesystem located on *device* to *pctfree*. Some systems also support an **-o** option to instruct the filesystem to optimize for fastest readback (**tunefs -o t** *device*) or least fragmentation (**tunefs -o s** *device*). When the reserve is pared to under 10%, fragmentation is usually more important.

tunefs alters information stored in a filesystem's superblock. Since the superblock is cached in memory while the filesystem is in use, **tunefs**

4. If root does use some extra space, **df** will report the disk as being more than 100% full.

should only be used on an unmounted filesystem. Otherwise, your changes will be clobbered the next time the cached superblock is written out. If you want to change the parameters of the root filesystem, run the **sync** command a couple of times, use **tunefs** to edit the superblock, and then do a **reboot -n** to reboot without any more **sync**s. This is best done from single-user mode.

HP-UX provides a nifty **-v** option to view a filesystem's current parameters. It also provides a **-A** option, which forces backup copies of the superblock to be edited as well as the master.

Most versions of the **tunefs** manual page mention that you can tune a filesystem, but you can't tune a fish. However, under OSF/1 you can apparently tune a fish.

BSDI's **dumpfs** command prints out the superblock of an existing filesystem. It can be used to examine the filesystem's state before you start making changes.

26.7 DISK QUOTAS

If you can keep your disk space under control using informal methods such as peer pressure and periodic audits, that is usually the best way to run the system. But if these techniques fail, you may need to install disk quotas to force your users to comply with "reasonable" limitations on their use of disk space.

Quotas allow you to limit the number of inodes and disk blocks that can be allocated to each user. The number of inodes roughly determines how many files a user can own. The disk block limitation controls the total amount of filesystem space a user can allocate.

Each limit is specified as two numbers: a *soft limit* after which the user is warned about the impending quota violation, and a *hard limit* that determines the absolute limit on the resource. Users are supposed to stay under their soft limits when not logged in.

To keep users from simply ignoring the soft limits, the quota system keeps track of how long someone has exceeded them. After a certain amount of time (often three days by default), the soft limit is enforced as rigidly as the hard limit, and nothing useful can be done until the user cleans up. At this point, the time limit is reset and the user can exceed the soft limits with impunity once again.

BSDI and OSF/1 support group quotas as well as user quotas, allowing you to limit the disk space consumed by a particular project or class of people. This is a nice feature because it gives you a way to protect different groups from each other's abuse while still leaving the space management within each group up to peer pressure and negotiation.

On systems without group quotas, disk partitioning can be used to achieve a similar effect. For example, separate partitions for students and faculty will stop faculty from taking so much space that students have none left. This provides rather gross control, since partitions are often a few hundred megabytes and house many individual users.

HP-UX provides only halfhearted support for disk partitioning, and does not support group quotas.

See Chapter 4, The Filesystem, for more information about ***chown****.*

Group quotas don't work well when users are in multiple groups; they can simply rotate among groups to evade the quotas. Another sneaky way for users to avoid quotas is to use the **chown** command to give away their files to other users. Quotas originated in BSD, which doesn't allow **chown** to be used by anyone but root. The ATT version of **chown** is a little different and does allow the owner of a file to give it away. In these days of Chinese-menu operating systems, it is not unheard of to see BSD's quotas and ATT's **chown** on the same system. There is usually a flag that can be set at kernel configuration time to disallow the use of **chown** by generic users.

HP-UX addresses the "**chown** vs. quotas" problem as part of a more general facility for controlling access to quasi-root-like privileges. Read the man page for **setprivgrp** for more details.

Quotas are useful not only to control true disk hogs, but also to stop runaway user programs that might otherwise fill the disk by mistake. Quotas provide a fine level of control, but require more maintenance than other means of controlling disk usage. Frantic users who cannot save their edit sessions become a constant administrative chore when disk space is tight.

See Chapter 29 for more information about disk performance.

Quotas can also reduce filesystem throughput by up to 30%. Since disk bandwidth has a dramatic effect on overall system performance, quotas can sometimes make a machine act sluggish. To minimize the performance cost of quotas, don't install them on the root partition, where most system-related disk activity occurs.

Quotas are handled on a per-user per-filesystem basis. If there is more than one filesystem on which a user is able to create files, quotas must be set for each one separately. If no quotas have been set for a particular user on a given filesystem, no default limit is applied. By convention, a limit of zero is also interpreted to mean "unlimited."

How Quotas Work

Quota information for a filesystem is kept in a file called **quotas** in the filesystem's root directory. The **quotas** file contains the limits placed on each user and also a summary of the amount of space consumed by each of the system's users.

On systems that support group quotas, there are two summary files: **quota.user** and **quota.group**. For brevity, we speak in this chapter as if there were only one file; you may have to repeat instructions that apply to the quota files if your system has group quotas.

The **edquota** command edits the per-user or per-group limits defined in the quota file or files. **quota** and **repquota** print out information about users' quotas and their current use of disk space.

The kernel normally updates the **quotas** file whenever filesystem operations change the number of disk blocks that a user is consuming. However, system crashes and other irregularities can introduce small errors into the summary file.

The **quotacheck** command examines a filesystem block by block to calculate the current disk usage, then updates the **quotas** file with an accurate summary. It is normally run with the **-a** flag at boot time, causing it to check every mounted filesystem declared to have quotas in the system's filesystem table (usually **/etc/fstab**). **quotacheck -v** prints a list of all users and their disk usages (a la **quot**). Most systems also understand **quotacheck -p**, which makes **quotacheck** examine filesystems in parallel in the manner of **fsck**.

HP-UX provides an even fancier **-P** option, which checks filesystems only when they appear to need it. This can save a lot of time.

The kernel doesn't automatically enforce quotas just because a filesystem contains a file called **quotas**; quotas have to be explicitly turned on after a filesystem is mounted using the **quotaon** command.

Enabling Quotas

Since quotas are a feature of the filesystem, they must be implemented in the kernel. Most systems today are shipped with quotas already enabled. Unfortunately, some are not—on these systems, you must build a new kernel that includes the quota code.

Chapter 13, *Configuring the Kernel*, describes the process of building a new kernel. To enable quotas, you usually add a line like

```
options QUOTA
```

to the new kernel's configuration file. After editing the config file, rebuild the kernel as described in Chapter 13 and reboot.

In addition to having quotas defined in the kernel, your system must explicitly start up quotas when the system boots. This involves running the **quotacheck -a -p** and **quotaon -a** commands[5] after local filesystems have been mounted (these commands do not work on unmounted

5. The **-a** flags apply the commands to all filesystems for which they are appropriate.

filesystems). This can be done from an **rc** startup script or from **/etc/inittab**, depending on how your system's startup routine works. Refer to Chapter 2, *Booting and Shutting Down*, for specifics.

A typical sequence of commands to mount, validate, and enable quotas on all filesystems is

```
/usr/etc/mount -a > /dev/console 2>&1
echo -n 'checking quotas:' >/dev/console
/usr/etc/quotacheck -a -p >/dev/console 2>&1
echo ' done.' >/dev/console
/usr/etc/quotaon -a
```

HP's version of **mount** understands quotas and turns them on automatically when a filesystem that supports them is mounted. HP-UX still provides the **quotaon** command, but it needn't be used at boot time.

Setting Up Quotas on a Particular Filesystem

Two steps must be taken to enable quotas on a filesystem. First, the **quotas** file must be created and configured, and second, the filesystem must be declared as using quotas in the filesystem table.

The **quotas** file should be owned by root. It should have read and write permissions for root and read permission for everyone else. For example, for a filesystem mounted as **/users**, you could create the **quotas** file with the following commands:

```
# touch /users/quotas
# chown root /users/quotas
# chmod 644 /users/quotas
```

These commands create an empty **quotas** file that you can populate with summary information by running **quotacheck** *devfile*, where *devfile* is the block device file (in **/dev**) on which the filesystem lives.

The filesystem table (usually **/etc/fstab**, sometimes **/etc/vfstab** or **/etc/checklist**) contains information about how disk partitions are set up and the uses to which they are to be put. Just as **mount** reads the table to find information about which filesystems should be mounted at startup time, **quotaon** and **quotacheck** read the table to find out which filesystems have quotas enabled.

Partitions on which you have not yet instituted quotas will normally have a line in the filesystem table that looks something like

```
/dev/ra0g /users      rw  1  2
```

The third field contains the code `rw`, meaning that the partition is to be mounted for both reading and writing (on your system, this field may contain additional options). To configure the filesystem for quotas, you

either replace `rw` with `rq` (read/write with quotas), or you add the additional option `quota`, depending on the system. See page 630 for platform-specific information.

Systems that provide both user and group quotas may use the `rq` convention; however, they also use options `userquota` and `groupquota` to explicitly enable each flavor of quota. Both of these options can take an argument indicating the pathname to the appropriate quota control file. The default names are **`quota.user`** and **`quota.group`**; there's usually no reason to change them.

Once you have set up a filesystem for quotas, you can reboot the system to make quotas take effect, or you can run **`quotaon`** *`filesystem`* to turn them on by hand. Since the format of the **`fstab`** file is rather picky, we recommend rebooting. A syntax error you introduce when enabling quotas may go undetected until a power failure months later.

edquota: Set Quotas

Once your filesystems are set up to support quotas, you can assign limits to specific users and groups with the **`edquota`** command.

`edquota` *`user`* will put you into **`vi`** (or whatever editor is specified in the EDITOR environment variable) to edit the quotas for that user on each mounted filesystem that is currently configured for quotas. You can specify multiple users, but since you will not be given any indication of which user's quotas you are editing at any given time, this is not very useful. The command

```
edquota -p proto-user user ...
```

sets *`user`*'s quotas to be the same as those of *`proto-user`*. Multiple users may be specified, but only one prototype user.

We maintain logins for a number of pseudo-users that are used only as disk quota prototypes. We used to have pseudo-users with several different quota allocations for each of about four different categories of account. This soon became too complex, so we simplified it to a three-level system: small, medium, and huge.

If your system supports group quotas, the command

```
edquota -g group
```

or the command

```
edquota -p proto-group -g group
```

will set the quotas for *`group`*. **`edquota -t`** is sets the amount of time that a user can stay over the soft quota limit before the system cracks down and enforces it. A variety of different time units are supported.

quota and repquota: View Quotas

To see the quotas set for a particular user, use the command

```
quota user
```

This command gives quota status information for filesystems on which *user* is over quota; complete information about quotas on all filesystems can be obtained by adding the **-v** flag. Individual users can find out their own quota information using the **quota** command, but only the superuser can see the quotas of other people.

login executes the **quota** command whenever a user logs in, thus warning the user about any quota problems that exist. This can cause annoying delays, especially with remote filesystems.

The **repquota** command produces a disk usage report similar to that of **quot** and **quotacheck -v**, but it also reports each user's quotas.

Quotas and NFS

See Chapter 17 for more information about the Network File System.

When filesystems are cross-mounted on a network, the implementation of quotas remains local to the machine that serves each filesystem. A client machine isn't responsible for doing any quota-related processing. But since quotas are checked on the server before each operation, quota limits are still enforced.

Most systems provide a simple daemon called **rquotad** which allows quota information to be queried over the network. It's used mostly to make the **quota** command work correctly for remote filesystems, so that users can be warned when they are over quota.

26.8 Disk Overflows

A filesystem that is completely full should be attended to as soon as possible. An overflow on a root, **/usr**, or **/var** partition is more important than one on a non-system partition, but no disk should be allowed to fester in an unusable state.

First, find out what caused the filesystem to overflow. If you have been keeping an eye on the filesystems and keeping them under 90% full, it is most likely some sort of runaway program that is filling the disk. Do a **ps** and look for suspicious processes. If you find the culprit, suspend it, inspect and possibly remove the files it was making, and apprise the process's owner of the situation.

If the overflow was not caused by a runaway program, you will need to remove some files to give breathing room for the filesystem until you can get people to clean up. If the overflowing filesystem is **/var**, remove whatever looks like junk in **/var/tmp** and truncate log files if you don't need to keep them for accounting. Check for kernel core dumps in the

/usr/crash or **/var/crash** directory if your system supports them; these can often be huge. If the problem is on a user filesystem, it may be harder to find things to delete, but core files are a good place to start.

You can use the **find** command to identify large files that have been recently created or modified. Here's a typical example:

```
find / -xdev -mtime -7 -size +200 -print
```

The exact syntax of the **find** command varies from system to system. This example is from SunOS; it lists files in the root partition that are larger than 100K and that have been modified in the last week. If you often have problems with overflowing disks, you might try running this command from **cron** every night and mailing yourself the results (perhaps with a higher size threshold).

26.9 SPECIFICS FOR VARIOUS OPERATING SYSTEMS

Quota-related commands are in **/usr/sbin**. Solaris does not support group quotas; quota information is kept in a single **quotas** file. The filesystem table is in **/etc/vfstab**. Use the `rq` option to request quotas on a particular filesystem.

Commands are in **/etc**. The mount table is **/etc/checklist**. Filesystems with quotas are marked with the option `quota`; `noquota` is also defined but is optional. HP-UX keeps track of the times when filesystems are gracefully unmounted, or when commands are used that might invalidate the quota summary information. This information is used to implement **quotacheck -P**, which acts like **quotacheck -p** but does no unnecessary work. **setprivgrp** can be used to set the behavior of the **chown** command. Group quotas are not supported.

Commands are in **/usr/etc**. Group quotas are not supported. The filesystem table is in **/etc/fstab**; use the `rq` option to request quotas on a particular filesystem. **quotacheck -n** *numusers* extends the size of the **quotas** file to accommodate *numusers* users. **quota -n** limits the display to local filesystems. IRIX does not have **tunefs**.

The filesystem table is **/etc/fstab**; `quota` and `noquota` options are used to turn quotas on or off. Commands are in **/usr/etc**. Group quotas are not supported.

Commands are in **/usr/sbin**. Group quotas are supported. The filesystem table is **/etc/fstab** and the `rq` option is used to request quotas, along with `userquota` and `groupquota` (see page 628).

Commands are in **/sbin**. The filesystem table is in **/etc/fstab**, and `userquota` and `groupquota` are used to enable user and group quotas. The `rq` option is not used. The **quot** command is not supported.