Disc Quotas in a UNIX* Environment

Robert Elz

Department of Computer Science
University of Melbourne,
Parkville,
Victoria,
Australia.

ABSTRACT

In most computing environments, disc space is not infinite. The disc quota system provides a mechanism to control usage of disc space, on an individual basis.

Quotas may be set for each individual user, on any, or all filesystems.

The quota system will warn users when they exceed their allotted limit, but allow some extra space for current work. Repeatedly remaining over quota at logout, will cause a fatal over quota condition eventually.

The quota system is an optional part of VMUNIX that may be included when the system is configured.

1. Users' view of disc quotas

To most users, disc quotas will either be of no concern, or a fact of life that cannot be avoided. The *quota* (1) command will provide information on any disc quotas that may have been imposed upon a user.

There are two individual possible quotas that may be imposed, usually if one is, both will be. A limit can be set on the amount of space a user can occupy, and there may be a limit on the number of files (inodes) he can own.

Quota provides information on the quotas that have been set by the system administrators, in each of these areas, and current usage.

There are four numbers for each limit, the current usage, soft limit (quota), hard limit, and number of remaining login warnings. The soft limit is the number of 1K blocks (or files) that the user is expected to remain below. Each time the user's usage goes past this limit, he will be warned. The hard limit cannot be exceeded. If a user's usage reaches this number, further requests for space (or attempts to create a file) will fail with an EDQUOT error, and the first time this occurs, a message will be written to the user's terminal. Only one message will be output, until space occupied is reduced below the limit, and reaches it again, in order to avoid continual noise from those programs that ignore write errors.

Whenever a user logs in with a usage greater than his soft limit, he will be warned, and his login warning count decremented. When he logs in under quota, the counter is reset to its maximum value (which is a system configuration parameter, that is typically 3). If the warning count should ever reach zero (caused by three successive logins over quota), the particular limit that has been exceeded will be treated as if the hard limit has been reached, and no more resources will be allocated to the user. The **only** way to reset this condition is to reduce usage below quota, then log in again.

1.1. Surviving when quota limit is reached

In most cases, the only way to recover from over quota conditions, is to abort whatever activity was in progress on the filesystem that has reached its limit, remove sufficient files to bring the limit back below quota, and retry the failed program.

However, if you are in the editor and a write fails because of an over quota situation, that is not a suitable course of action, as it is most likely that initially attempting to write the file will have truncated its previous contents, so should the editor be aborted without correctly writing the file not only will the recent changes be lost, but possibly much, or even all, of the data that previously existed.

There are several possible safe exits for a user caught in this situation. He may use the editor **!** shell escape command to examine his file space, and remove surplus files. Alternatively, using *csh*, he may suspend the editor, remove some files, then resume it. A third possibility, is to write the file to some other filesystem (perhaps to a file on /tmp) where the user's quota has not been exceeded. Then after rectifying the quota situation, the file can be moved back to the filesystem it belongs on.

2. Administering the quota system

To set up and establish the disc quota system, there are several steps necessary to be performed by the system administrator.

First, the system must be configured to include the disc quota sub-system. This is done by including the line:

```
options   QUOTA
```

in the system configuration file, then running *config* (8) followed by a system configuration*.

Second, a decision as to what filesystems need to have quotas applied needs to be made. Usually, only filesystems that house users' home directories, or other user files, will need to be subjected to the quota system, though it may also prove useful to also include **/usr**. If possible, **/tmp** should usually be free of quotas.

Having decided on which filesystems quotas need to be set upon, the administrator should then allocate the available space amongst the competing needs. How this should be done is (way) beyond the scope of this document.

Then, the *edquota* (8) command can be used to actually set the limits desired upon each user. Where a number of users are to be given the same quotas (a common occurrence) the **−p** switch to edquota will allow this to be easily accomplished.

Once the quotas are set, ready to operate, the system must be informed to enforce quotas on the desired filesystems. This is accomplished with the *quotaon* (8) command. *Quotaon* will either enable quotas for a particular filesystem, or with the **−a** switch, will enable quotas for each filesystem indicated in **/etc/fstab** as using quotas. See *fstab* (5) for details. Most sites using the quota system, will include the line

```
/etc/quotaon -a
```

in **/etc/rc.local**.

Should quotas need to be disabled, the *quotaoff*(8) command will do that, however, should the filesystem be about to be dismounted, the *umount* (8) command will disable quotas immediately before the filesystem is unmounted. This is actually an effect of the *umount* (2) system call, and it guarantees that the quota system will not be disabled if the umount would fail because the filesystem is not idle.

Periodically (certainly after each reboot, and when quotas are first enabled for a filesystem), the records retained in the quota file should be checked for consistency with the actual number of blocks and files allocated to the user. The *quotacheck* (8) command can be used to accomplish this. It is not necessary to dismount the filesystem, or disable the quota system to run this command, though on active filesystems inaccurate results may occur. This does no real harm in most cases, another run of *quotacheck* when the filesystem is idle will certainly correct any inaccuracy.

* See also the document ''Building 4.2BSD UNIX Systems with Config''.

The super-user may use the *quota* (1) command to examine the usage and quotas of any user, and the *repquota* (8) command may be used to check the usages and limits for all users on a filesystem.

3. Some implementation detail.

Disc quota usage and information is stored in a file on the filesystem that the quotas are to be applied to. Conventionally, this file is **quotas** in the root of the filesystem. While this name is not known to the system in any way, several of the user level utilities "know" it, and choosing any other name would not be wise.

The data in the file comprises an array of structures, indexed by uid, one structure for each user on the system (whether the user has a quota on this filesystem or not). If the uid space is sparse, then the file may have holes in it, which would be lost by copying, so it is best to avoid this.

The system is informed of the existence of the quota file by the *setquota* (2) system call. It then reads the quota entries for each user currently active, then for any files open owned by users who are not currently active. Each subsequent open of a file on the filesystem, will be accompanied by a pairing with its quota information. In most cases this information will be retained in core, either because the user who owns the file is running some process, because other files are open owned by the same user, or because some file (perhaps this one) was recently accessed. In memory, the quota information is kept hashed by user-id and filesystem, and retained in an LRU chain so recently released data can be easily reclaimed. Information about those users whose last process has recently terminated is also retained in this way.

Each time a block is accessed or released, and each time an inode is allocated or freed, the quota system gets told about it, and in the case of allocations, gets the opportunity to object.

Measurements have shown that the quota code uses a very small percentage of the system cpu time consumed in writing a new block to disc.

4. Acknowledgments

The current disc quota system is loosely based upon a very early scheme implemented at the University of New South Wales, and Sydney University in the mid 70's. That system implemented a single combined limit for both files and blocks on all filesystems.

A later system was implemented at the University of Melbourne by the author, but was not kept highly accurately, eg: chown's (etc) did not affect quotas, nor did i/o to a file other than one owned by the instigator.

The current system has been running (with only minor modifications) since January 82 at Melbourne. It is actually just a small part of a much broader resource control scheme, which is capable of controlling almost anything that is usually uncontrolled in unix. The rest of this is, as yet, still in a state where it is far too subject to change to be considered for distribution.

For the 4.2BSD release, much work has been done to clean up and sanely incorporate the quota code by Sam Leffler and Kirk McKusick at The University of California at Berkeley.