toolman

Meet Ed Q.

by Daniel E. Singer
<*des@cs.duke.edu*>

Dan has been doing a mix of programming and system administration since 1983. He is currently a system administrator in the Duke University Department of Computer Science in Durham, North Carolina.

Modifying UNIX disk quotas can be a hassle. The command interface provided by major UNIX platforms[1] is quite cumbersome. Perhaps this issue will be addressed (fixed) one day, with appropriate command-line options or some other mechanism being added to the API.

In this article I'll discuss a tool that helps to work around this unnecessarily annoying system-administration duty. Some basic knowledge of the standard UNIX disk-quota system is assumed.

Background

In the course of system-administration activities, it's not uncommon to have to create or maintain user disk quotas as a component of user-account and disk-space management. It's nice to have a quick and efficient means for making these routine updates. And if you use any type of script or program for automating the setup of accounts, it's also nice to be able to automate quota setup as part of the process.

The facility provided by most UNIX platforms for editing user disk quotas in the standard file system is the `edquota` command. `edquota` allows you to create or modify user quotas one at a time, either via an interactive text-editor interface or via a prototype. The text-editor method allows flexibility, but it is time-consuming and not readily adaptable to automation. The prototype method can be automated (i.e., used noninteractively), but it requires you to have preexisting prototypes that can only be copied exactly; various problems are inherent in this. See `man edquota` for more details.

Moving Forward

The shortcomings of `edquota` make an alternative method seem desirable. One possibility is to write a C program using quota-related system calls. This is not a bad choice, but it involves the usual hassles of maintaining a compiled program, and possibly needing distinct binaries for each of multiple UNIX platforms. For us script-heads, a script approach is much more attractive.

Several years ago, a coworker mentioned having seen a USENET posting[2] that demonstrated a clever use of the EDITOR environment variable with `edquota`, and he suggested that this mechanism might be exploited to impose some automation on the quota-editing process. After some consideration, this approach did appear to me to be workable. You see, `edquota` invokes the editor of your choice to modify a quota, this *usually* being an interactive text editor.

So the basic plan followed is this: a Bourne shell script processes command-line arguments (user, partition, quota, options, etc.), sets the EDITOR environment variable to some program, then runs `edquota`. The program that EDITOR points to automatically (noninteractively) does the quota editing.[3] In this implementation, the script and the

EDITOR program are actually the same script, just called by different names (via a hard or symbolic link). So, `edq`, our script, calls `edquota`, which then calls `edq` (by a different, linked name) to do the quota editing.[4]

Execution

A typical use of `edq` (pronounced "ed cue") looks something like this:

```
# edq tina,,100000
```

This invocation sets the home directory blocks soft limit for the user account "tina" to 100,000 KB, the blocks hard limit to 125,000 KB, the inodes soft limit to 20,000, and the inodes hard limit to 25,000. Apparently some defaults were involved in this first example.

Here's a more explicit (and perhaps more general) example:

```
# edq adam,/work1,50000,10000 betti,/work2,80000,20000
```

Here we've set the quotas for two accounts at once, and we've also specified the partitions and soft limits for blocks and inodes explicitly. (The hard limits are still being set by hard-coded default ratios; see below.)

To summarize a bit, `edq` can noninteractively set one or more disk quotas for one or more users on one or more file systems, optionally employing various defaults. If the partition is omitted, the home directory is assumed. If the inode limit is omitted, it is computed from a ratio to the blocks limit. Hard limits are based on soft limits, and are also computed from ratios to the soft limits. Defaults for all of these ratios can easily be set in the script, but can all be overridden with environment variables:

- INODES_SOFT_RATIO: ratio of inodes to number of blocks
- BLOCKS_HARD_RATIO: ratio of blocks hard limit to soft limit
- INODES_HARD_RATIO: ratio of inodes hard limit to soft limit

(Run `edq -h` to see the defaults.)

Remote Control

What if the disk partition in question is on another host? Not to worry! Either the partition can be specified with a *host:/filesystem* notation, or the -r (remote) flag can be used. So, let's say you have a script that sets up accounts. The home directory quota could be set with a line similar to this:

```
INODES_HARD_RATIO=1.2 INODES_SOFT_RATIO=.3 \
BLOCKS_HARD_RATIO=1.2 edq -r $user,,$blocks
```

This sets the blocks soft limit for `$user` to `$blocks`, sets the inodes soft limit to `($blocks * .3)`, sets the blocks hard limit to `($blocks * 1.2)`, and sets the inodes hard limit to `($blocks * .3 * 1.2)`, assuming appropriate permissions to the possibly remote host housing the home directory partition for `$user`. (`rsh` could also be used for the remote access, but the remote host name would have to be provided; `-r` will figure out the right host.) So besides the user, the only other piece of hard information needed is the blocks soft limit.

Why this ratio approach? I don't want to have to figure out or maintain a table of all of the different values for each quota I might use for different kinds of accounts or for different partitions. So I just use ratios from what seems to me to be the most obvious figure, the blocks soft quota. For my purposes, having the inodes soft limit be 20% of the blocks limit, and then having the hard limits be 25% greater than the soft limits, seems about right, and these figures are hard-coded into edq.

Admittedly, the environment variable, ratios, etc., might seem a bit klutzy. But I've tried to keep the option list simple and allow for multiple quotas to be set with one call (the latter admittedly being a feature I don't use very often). Since it is a script, you can easily edit it and reset the default ratios to your preferences, and then never have

to worry about it again. I have it set to my preferences, and I never seem to have any need to deviate. Plus, it is designed so that use of the defaults can usually be leveraged, as in the very first example above. The use of commas for a compound argument delimiter also helps make it easier to omit components so as to go with defaults.

Syntax Variations

As I've mentioned, allowing for setting multiple quotas on the command line is perhaps dubious. I've considered whether it would make more sense just to allow for multiple account names or partitions, with the other data being specified only once. But this is not something one would need very often, except possibly for multiple account names if you are setting up multiple user accounts simultaneously, or when applying quotas to a new partition for the first time. But, then, I figure why bother, since these can be accomplished from the command line using the macro expansions supplied with many modern shells. For example, with the `csh`derivatives:

```
# edq -r {amy,barney,cathie},,60000
```

would automatically expand out to:

```
# edq -r amy,,60000 barney,,60000 cathie,,60000
```

And so it is already taken care of.

Other Languages

As I mentioned earlier, I'd prefer not to use a compiled language for this tool. Other scripting languages could probably accomplish this task with less code and more efficiency. For example, here's some Perl code provided by Seth Vidal of the Duke University physics department:

```
#!/usr/bin/perl
use Quota;
if (scalar(@ARGV) < 6) {
        print "\nUsage: setquota.pl mountpoint username blocksoft blockhard
filesoft filehard\n\n";
        exit(1);
}
my ($dir,$user,$bs,$bh,$is,$ih) = @ARGV;
$dev=Quota::getqcarg("$dir");
my @userstuff=getpwnam("$user");
my $uid=$userstuff[2];
Quota::setqlim($dev,$uid,$bs,$bh,$is,$ih,1);
```

In comparison, version 1.7 of `edq` is 837 lines long. But then, `edq` does contain some comments and a few more features. I chose Bourne shell because I like it, it's fairly portable, it's ubiquitous, and, well, because I wanted to try out that EDITOR environment variable trick suggested by that USENET posting. And, hell, it works!

Summary

The ability to easily update user disk quotas noninteractively has been overlooked for far too long by UNIX vendors and standards bodies. `edq` is one attempt to address this problem, providing a simple command-line interface for setting disk quotas both locally and remotely, with various details being rationally handled by practical defaults.

`edq` has been tested on fairly modern versions of Solaris and Digital UNIX, and also on SunOS. Some modifications will be required for other UNIXes. Drop me a line if you need help with a port, or if you want to provide one.

The `edq` Bourne shell script and its man page can be found at either of these addresses:
<*http://www.cs.duke.edu/toolman/*>
<*ftp://ftp.cs.duke.edu/pub/des/scripts/*>

NOTES

1. AIX, BSDs, D.U., HP-UX, Irix, Linux, Solaris, and SunOS, to name a few.

2. The posting in question — by the venerable Casper Dik in <*comp.sys.sun.admin*>, dated 14 Feb 1996 10:14:46 GMT — actually presented a method of adding a SUID wrapper to`edquota`; this particular issue is not addressed in this article.

3. The way this actually works is that `edquota` dumps some data into a temporary file, and then calls `$EDITOR` with the temporary file pathname as an argument; for example, vi /tmp/EdP.aXyyxV. After `$EDITOR` exits, `edquota` reads the temporary file back in and continues from there.

4. This is similar to the approach for using the EDITOR environment variable and the rc script as a revision-control wrapper around some other process. See the "Sleight of Hand" section in "Revision Control Revisited" in the October 1999 issue of *;login:*.