

ulimit and sysctl

The ulimit and sysctl programs allow to limit system-wide resource use. This can help a lot in system administration, e.g. when a user starts too many processes and therefore makes the system unresponsive for other users.

Code Listing 1: ulimit example

```
# ulimit -a
core file size          (blocks, -c) 0
data seg size          (kbytes, -d) unlimited
file size              (blocks, -f) unlimited
pending signals        (-i) 8191
max locked memory      (kbytes, -l) 32
max memory size        (kbytes, -m) unlimited
open files             (-n) 1024
pipe size              (512 bytes, -p) 8
POSIX message queues   (bytes, -q) 819200
stack size             (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes     (-u) 8191
virtual memory         (kbytes, -v) unlimited
file locks             (-x) unlimited
```

All these settings can be manipulated. A good example is this bash forkbomb that forks as many processes as possible and can crash systems where no user limits are set:

Warning: Do not run this in a shell! If no limits are set your system will either become unresponsive or might even crash.

Code Listing 2: A bash forkbomb

```
$ :(){ :|:& };:
```

Now this is not good - any user with shell access to your box could take it down. But if that user can only start 30 processes the damage will be minimal. So let's set a process limit:

Gentoo Note: A too small number of processes can break the use of portage. So, don't be too strict.

Code Listing 3: Setting a process limit

```
# ulimit -u 30
# ulimit -a
â€¦|
max user processes     (-u) 30
â€¦|
```

If you try to run the forkbomb now it should run, but throw error messages "fork: resource temporarily unavailable". This means that your system has not allowed the forkbomb to start more processes. The other options of ulimit can help with similar problems, but you should be careful that you don't lock yourself out - setting data seg size too small will even prevent bash from starting!

sysctl is a similar tool: It allows to configure kernel parameters at runtime. If you wish to keep settings persistent across reboots you should edit `/etc/sysctl.conf` - be aware that wrong settings may break things in unforeseen ways.

Code Listing 4: Exploring sysctl variables

```
# sysctl -a
vm.swappiness = 60
```

The list of variables is quite long (367 lines on my system), but I picked out `vm.swappiness` here. It controls how aggressive swapping will be, the higher it is (with a maximum of 100) the more swap will be used. This can affect performance a lot on systems with little memory, depending on load and other factors.

Code Listing 5: Reducing swappiness

```
# sysctl vm.swappiness=0
vm.swappiness = 0
```

The effects of changing this setting are usually not felt instantly. But you can change many settings, especially network-related, this way. For servers this can offer a nice performance boost, but as with `ulimit` careless usage might cause your system to misbehave or slow down. If you don't know what a variable controls, you should not modify it!

from <http://www.gentoo.org/news/en/gwn/20050808-newsletter.xml>