

Documentation for /proc/sys/fs/* kernel version 2.2.10
 (c) 1998, 1999, Rik van Riel <riel@nl.linux.org>
 (c) 2009, Shen Feng<shen@cn.fujitsu.com>

For general info and legal blurb, please look in README.

=====

This file contains documentation for the sysctl files in
 /proc/sys/fs/ and is valid for Linux kernel version 2.2.

The files in this directory can be used to tune and monitor
 miscellaneous and general things in the operation of the Linux
 kernel. Since some of the files can be used to screw up your
 system, it is advisable to read both documentation and source
 before actually making adjustments.

1. /proc/sys/fs

=====

Currently, these files are in /proc/sys/fs:

- aio-max-nr
- aio-nr
- dentry-state
- dquot-max
- dquot-nr
- file-max
- file-nr
- inode-max
- inode-nr
- inode-state
- nr_open
- overflowuid
- overflowgid
- suid_dumpable
- super-max
- super-nr

=====

aio-nr & aio-max-nr:

aio-nr is the running total of the number of events specified on the
 io_setup system call for all currently active aio contexts. If aio-nr
 reaches aio-max-nr then io_setup will fail with EAGAIN. Note that
 raising aio-max-nr does not result in the pre-allocation or re-sizing
 of any kernel data structures.

=====

dentry-state:

From linux/fs/dentry.c:

=====

```
struct {
    int nr_dentry;
```

fs.txt.txt

```
int nr_unused;
int age_limit;      /* age in seconds */
int want_pages;     /* pages requested by system */
int dummy[2];
} dentry_stat = {0, 0, 45, 0,};
```

Dentries are dynamically allocated and deallocated, and `nr_dentry` seems to be 0 all the time. Hence it's safe to assume that only `nr_unused`, `age_limit` and `want_pages` are used. `Nr_unused` seems to be exactly what its name says. `Age_limit` is the age in seconds after which dcache entries can be reclaimed when memory is short and `want_pages` is nonzero when `shrink_dcache_pages()` has been called and the dcache isn't pruned yet.

dquot-max & dquot-nr:

The file `dquot-max` shows the maximum number of cached disk quota entries.

The file `dquot-nr` shows the number of allocated disk quota entries and the number of free disk quota entries.

If the number of free cached disk quotas is very low and you have some awesome number of simultaneous system users, you might want to raise the limit.

file-max & file-nr:

The kernel allocates file handles dynamically, but as yet it doesn't free them again.

The value in `file-max` denotes the maximum number of file handles that the Linux kernel will allocate. When you get lots of error messages about running out of file handles, you might want to increase this limit.

Historically, the three values in `file-nr` denoted the number of allocated file handles, the number of allocated but unused file handles, and the maximum number of file handles. Linux 2.6 always reports 0 as the number of free file handles -- this is not an error, it just means that the number of allocated file handles exactly matches the number of used file handles.

Attempts to allocate more file descriptors than `file-max` are reported with `printk`, look for "VFS: file-max limit <number> reached".

`nr_open`:

fs.txt.txt

This denotes the maximum number of file-handles a process can allocate. Default value is 1024*1024 (1048576) which should be enough for most machines. Actual limit depends on RLIMIT_NOFILE resource limit.

=====

inode-max, inode-nr & inode-state:

As with file handles, the kernel allocates the inode structures dynamically, but can't free them yet.

The value in inode-max denotes the maximum number of inode handlers. This value should be 3-4 times larger than the value in file-max, since stdin, stdout and network sockets also need an inode struct to handle them. When you regularly run out of inodes, you need to increase this value.

The file inode-nr contains the first two items from inode-state, so we'll skip to that file...

Inode-state contains three actual numbers and four dummies. The actual numbers are, in order of appearance, nr_inodes, nr_free_inodes and preshrink.

Nr_inodes stands for the number of inodes the system has allocated, this can be slightly more than inode-max because Linux allocates them one pageful at a time.

Nr_free_inodes represents the number of free inodes (?) and preshrink is nonzero when the nr_inodes > inode-max and the system needs to prune the inode list instead of allocating more.

=====

overflowgid & overflowuid:

Some filesystems only support 16-bit UIDs and GIDs, although in Linux UIDs and GIDs are 32 bits. When one of these filesystems is mounted with writes enabled, any UID or GID that would exceed 65535 is translated to a fixed value before being written to disk.

These sysctls allow you to change the value of the fixed UID and GID. The default is 65534.

=====

suid_dumpable:

This value can be used to query and set the core dump mode for setuid or otherwise protected/tainted binaries. The modes are

- 0 - (default) - traditional behaviour. Any process which has changed privilege levels or is execute only will not be dumped
- 1 - (debug) - all processes dump core when possible. The core dump is

fs.txt.txt

owned by the current user and no security is applied. This is intended for system debugging situations only. Ptrace is unchecked.
2 - (suidsafe) - any binary which normally would not be dumped is dumped readable by root only. This allows the end user to remove such a dump but not access it directly. For security reasons core dumps in this mode will not overwrite one another or other files. This mode is appropriate when administrators are attempting to debug problems in a normal environment.

=====

super-max & super-nr:

These numbers control the maximum number of superblocks, and thus the maximum number of mounted filesystems the kernel can have. You only need to increase super-max if you need to mount more filesystems than the current value in super-max allows you to.

=====

aio-nr & aio-max-nr:

aio-nr shows the current system-wide number of asynchronous io requests. aio-max-nr allows you to change the maximum value aio-nr can grow to.

=====

2. /proc/sys/fs/binfmt_misc

Documentation for the files in /proc/sys/fs/binfmt_misc is in Documentation/binfmt_misc.txt.

3. /proc/sys/fs/mqueue - POSIX message queues filesystem

The "mqueue" filesystem provides the necessary kernel features to enable the creation of a user space library that implements the POSIX message queues API (as noted by the MSG tag in the POSIX 1003.1-2001 version of the System Interfaces specification.)

The "mqueue" filesystem contains values for determining/setting the amount of resources used by the file system.

/proc/sys/fs/mqueue/queues_max is a read/write file for setting/getting the maximum number of message queues allowed on the system.

/proc/sys/fs/mqueue/msg_max is a read/write file for setting/getting the maximum number of messages in a queue value. In fact it is the limiting value for another (user) limit which is set in mq_open invocation. This attribute of a queue must be less or equal then msg_max.

fs.txt.txt

/proc/sys/fs/mqueue/msgsize_max is a read/write file for setting/getting the maximum message size value (it is every message queue's attribute set during its creation).

4. /proc/sys/fs/epoll - Configuration options for the epoll interface

This directory contains configuration options for the epoll(7) interface.

max_user_instances

This is the maximum number of epoll file descriptors that a single user can have open at a given time. The default value is 128, and should be enough for normal users.

max_user_watches

Every epoll file descriptor can store a number of files to be monitored for event readiness. Each one of these monitored files constitutes a "watch". This configuration option sets the maximum number of "watches" that are allowed for each user.

Each "watch" costs roughly 90 bytes on a 32bit kernel, and roughly 160 bytes on a 64bit one.

The current default value for max_user_watches is the 1/32 of the available low memory, divided for the "watch" cost in bytes.