proc(5) — Linux manual page

**PROC(5) Linux Programmer's Manual**

proc - process information pseudo-filesystem

**DESCRIPTION**[**top**](https://man7.org/linux/man-pages/man5/proc.5.html#top_of_page)

The **proc** filesystem is a pseudo-filesystem which provides an

interface to kernel data structures. It is commonly mounted at

*/proc*. Typically, it is mounted automatically by the system, but

it can also be mounted manually using a command such as:

mount -t proc proc /proc

Most of the files in the **proc** filesystem are read-only, but some

files are writable, allowing kernel variables to be changed.

**Mount options**

The **proc** filesystem supports the following mount options:

**hidepid**=*n* (since Linux 3.3)

This option controls who can access the information in

*/proc/[pid]* directories. The argument, *n*, is one of the

following values:

0 Everybody may access all */proc/[pid]* directories.

This is the traditional behavior, and the default if

this mount option is not specified.

1 Users may not access files and subdirectories inside

any */proc/[pid]* directories but their own (the

*/proc/[pid]* directories themselves remain visible).

Sensitive files such as */proc/[pid]/cmdline* and

*/proc/[pid]/status* are now protected against other

users. This makes it impossible to learn whether any

user is running a specific program (so long as the

program doesn't otherwise reveal itself by its

behavior).

2 As for mode 1, but in addition the */proc/[pid]*

directories belonging to other users become invisible.

This means that */proc/[pid]* entries can no longer be

used to discover the PIDs on the system. This doesn't

hide the fact that a process with a specific PID value

exists (it can be learned by other means, for example,

by "kill -0 $PID"), but it hides a process's UID and

GID, which could otherwise be learned by employing

[stat(2)](https://man7.org/linux/man-pages/man2/stat.2.html) on a */proc/[pid]* directory. This greatly

complicates an attacker's task of gathering

information about running processes (e.g., discovering

whether some daemon is running with elevated

privileges, whether another user is running some

sensitive program, whether other users are running any

program at all, and so on).

**gid**=*gid* (since Linux 3.3)

Specifies the ID of a group whose members are authorized

to learn process information otherwise prohibited by

**hidepid** (i.e., users in this group behave as though */proc*

was mounted with *hidepid=0*). This group should be used

instead of approaches such as putting nonroot users into

the [sudoers(5)](https://man7.org/linux/man-pages/man5/sudoers.5.html) file.

**Overview**

Underneath */proc*, there are the following general groups of files

and subdirectories:

*/proc/[pid]* subdirectories

Each one of these subdirectories contains files and

subdirectories exposing information about the process with

the corresponding process ID.

Underneath each of the */proc/[pid]* directories, a *task*

subdirectory contains subdirectories of the form

*task/[tid]*, which contain corresponding information about

each of the threads in the process, where *tid* is the

kernel thread ID of the thread.

The */proc/[pid]* subdirectories are visible when iterating

through */proc* with [getdents(2)](https://man7.org/linux/man-pages/man2/getdents.2.html) (and thus are visible when

one uses [ls(1)](https://man7.org/linux/man-pages/man1/ls.1.html) to view the contents of */proc*).

*/proc/[tid]* subdirectories

Each one of these subdirectories contains files and

subdirectories exposing information about the thread with

the corresponding thread ID. The contents of these

directories are the same as the corresponding

*/proc/[pid]/task/[tid]* directories.

The */proc/[tid]* subdirectories are *not* visible when

iterating through */proc* with [getdents(2)](https://man7.org/linux/man-pages/man2/getdents.2.html) (and thus are *not*

visible when one uses [ls(1)](https://man7.org/linux/man-pages/man1/ls.1.html) to view the contents of

*/proc*).

*/proc/self*

When a process accesses this magic symbolic link, it

resolves to the process's own */proc/[pid]* directory.

*/proc/thread-self*

When a thread accesses this magic symbolic link, it

resolves to the process's own */proc/self/task/[tid]*

directory.

*/proc/[a-z]\**

Various other files and subdirectories under */proc* expose

system-wide information.

All of the above are described in more detail below.

**Files and directories**

The following list provides details of many of the files and

directories under the */proc* hierarchy.

*/proc/[pid]*

There is a numerical subdirectory for each running

process; the subdirectory is named by the process ID.

Each */proc/[pid]* subdirectory contains the pseudo-files

and directories described below.

The files inside each */proc/[pid]* directory are normally

owned by the effective user and effective group ID of the

process. However, as a security measure, the ownership is

made *root:root* if the process's "dumpable" attribute is

set to a value other than 1.

Before Linux 4.11, *root:root* meant the "global" root user

ID and group ID (i.e., UID 0 and GID 0 in the initial user

namespace). Since Linux 4.11, if the process is in a

noninitial user namespace that has a valid mapping for

user (group) ID 0 inside the namespace, then the user

(group) ownership of the files under */proc/[pid]* is

instead made the same as the root user (group) ID of the

namespace. This means that inside a container, things

work as expected for the container "root" user.

The process's "dumpable" attribute may change for the

following reasons:

\* The attribute was explicitly set via the [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html)

**PR\_SET\_DUMPABLE** operation.

\* The attribute was reset to the value in the file

*/proc/sys/fs/suid\_dumpable* (described below), for the

reasons described in [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html).

Resetting the "dumpable" attribute to 1 reverts the

ownership of the */proc/[pid]/\** files to the process's

effective UID and GID. Note, however, that if the

effective UID or GID is subsequently modified, then the

"dumpable" attribute may be reset, as described in

[prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html). Therefore, it may be desirable to reset the

"dumpable" attribute *after* making any desired changes to

the process's effective UID or GID.

*/proc/[pid]/attr*

The files in this directory provide an API for security

modules. The contents of this directory are files that

can be read and written in order to set security-related

attributes. This directory was added to support SELinux,

but the intention was that the API be general enough to

support other security modules. For the purpose of

explanation, examples of how SELinux uses these files are

provided below.

This directory is present only if the kernel was

configured with **CONFIG\_SECURITY**.

Файлы в этом каталоге предоставляют API для обеспечения безопасности.

модули. Содержимое этого каталога — файлы, которые

могут быть прочитаны и записаны, чтобы установить связанные с безопасностью

атрибуты. Этот каталог был добавлен для поддержки SELinux,

но намерение состояло в том, чтобы API был достаточно общим, чтобы

поддержка других модулей безопасности. С целью

объяснение, примеры того, как SELinux использует эти файлы,

предоставлено ниже.

Этот каталог присутствует, только если ядро было

настроен с помощью CONFIG\_SECURITY.

*/proc/[pid]/attr/current* (since Linux 2.6.0)

The contents of this file represent the current security

attributes of the process.

In SELinux, this file is used to get the security context

of a process. Prior to Linux 2.6.11, this file could not

be used to set the security context (a write was always

denied), since SELinux limited process security

transitions to [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html) (see the description of

*/proc/[pid]/attr/exec*, below). Since Linux 2.6.11,

SELinux lifted this restriction and began supporting "set"

operations via writes to this node if authorized by

policy, although use of this operation is only suitable

for applications that are trusted to maintain any desired

separation between the old and new security contexts.

Prior to Linux 2.6.28, SELinux did not allow threads

within a multithreaded process to set their security

context via this node as it would yield an inconsistency

among the security contexts of the threads sharing the

same memory space. Since Linux 2.6.28, SELinux lifted

this restriction and began supporting "set" operations for

threads within a multithreaded process if the new security

context is bounded by the old security context, where the

bounded relation is defined in policy and guarantees that

the new security context has a subset of the permissions

of the old security context.

Other security modules may choose to support "set"

operations via writes to this node.

Содержимое этого файла представляет текущую безопасность

атрибуты процесса.

В SELinux этот файл используется для получения контекста безопасности.

процесса. До Linux 2.6.11 этот файл не мог

использоваться для установки контекста безопасности (запись всегда

отклонено), так как SELinux ограничивает безопасность процесса

переходы в execve(2) (см. описание

/proc/[pid]/attr/exec, ниже). Начиная с Linux 2.6.11,

SELinux снял это ограничение и начал поддерживать «set».

операции через запись в этот узел, если это разрешено

политики, хотя использование этой операции подходит только

для приложений, которым доверяют поддерживать любые желаемые

разделение старого и нового контекстов безопасности.

До Linux 2.6.28 SELinux не разрешал потоки.

в многопоточном процессе, чтобы установить их безопасность

контекст через этот узел, так как это приведет к несогласованности

среди контекстов безопасности потоков, разделяющих

одно и то же пространство памяти. Начиная с Linux 2.6.28, SELinux снят

это ограничение и начал поддерживать операции "set" для

потоки внутри многопоточного процесса, если новая безопасность

контекст ограничен старым контекстом безопасности, где

ограниченное отношение определяется в политике и гарантирует, что

новый контекст безопасности имеет подмножество разрешений

старого контекста безопасности.

Другие модули безопасности могут выбрать поддержку «set».

операции через запись в этот узел.

*/proc/[pid]/attr/exec* (since Linux 2.6.0)

This file represents the attributes to assign to the

process upon a subsequent [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html).

In SELinux, this is needed to support role/domain

transitions, and [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html) is the preferred point to make

such transitions because it offers better control over the

initialization of the process in the new security label

and the inheritance of state. In SELinux, this attribute

is reset on [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html) so that the new program reverts to

the default behavior for any [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html) calls that it may

make. In SELinux, a process can set only its own

*/proc/[pid]/attr/exec* attribute.

*/proc/[pid]/attr/fscreate* (since Linux 2.6.0)

This file represents the attributes to assign to files

created by subsequent calls to [open(2)](https://man7.org/linux/man-pages/man2/open.2.html), [mkdir(2)](https://man7.org/linux/man-pages/man2/mkdir.2.html),

[symlink(2)](https://man7.org/linux/man-pages/man2/symlink.2.html), and [mknod(2)](https://man7.org/linux/man-pages/man2/mknod.2.html)

SELinux employs this file to support creation of a file

(using the aforementioned system calls) in a secure state,

so that there is no risk of inappropriate access being

obtained between the time of creation and the time that

attributes are set. In SELinux, this attribute is reset

on [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html), so that the new program reverts to the

default behavior for any file creation calls it may make,

but the attribute will persist across multiple file

creation calls within a program unless it is explicitly

reset. In SELinux, a process can set only its own

*/proc/[pid]/attr/fscreate* attribute.

Этот файл представляет атрибуты для назначения

процесс при последующем execve(2).

В SELinux это необходимо для поддержки роли/домена.

переходы, и execve(2) является предпочтительной точкой для выполнения

такие переходы, потому что он предлагает лучший контроль над

инициализация процесса в новой метке безопасности

и наследование государства. В SELinux этот атрибут

сбрасывается в execve(2), так что новая программа возвращается к

поведение по умолчанию для любых вызовов execve(2), которые он может

сделать. В SELinux процесс может устанавливать только свои

/proc/[pid]/attr/exec атрибут.

/proc/[pid]/attr/fscreate (начиная с Linux 2.6.0)

Этот файл представляет атрибуты для присвоения файлам

созданные последующими вызовами open(2), mkdir(2),

символическая ссылка (2) и mknod (2)

SELinux использует этот файл для поддержки создания файла

(используя вышеупомянутые системные вызовы) в безопасном состоянии,

чтобы не было риска неправомерного доступа

полученный между временем создания и временем, когда

установлены атрибуты. В SELinux этот атрибут сбрасывается

на execve(2), чтобы новая программа вернулась к

поведение по умолчанию для любых вызовов создания файлов, которые он может сделать,

но атрибут будет сохраняться в нескольких файлах

вызовы создания внутри программы, если это явно не указано

сброс настроек. В SELinux процесс может устанавливать только свои

/proc/[pid]/attr/fscreate атрибут.

*/proc/[pid]/attr/keycreate* (since Linux 2.6.18)

If a process writes a security context into this file, all

subsequently created keys ([add\_key(2)](https://man7.org/linux/man-pages/man2/add_key.2.html)) will be labeled

with this context. For further information, see the

kernel source file *Documentation/security/keys/core.rst*

(or file *Documentation/security/keys.txt* on Linux between

3.0 and 4.13, or *Documentation/keys.txt* before Linux 3.0).

*/proc/[pid]/attr/prev* (since Linux 2.6.0)

This file contains the security context of the process

before the last [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html); that is, the previous value of

*/proc/[pid]/attr/current*.

*/proc/[pid]/attr/socketcreate* (since Linux 2.6.18)

If a process writes a security context into this file, all

subsequently created sockets will be labeled with this

context.

*/proc/[pid]/autogroup* (since Linux 2.6.38)

See [sched(7)](https://man7.org/linux/man-pages/man7/sched.7.html).

*/proc/[pid]/auxv* (since 2.6.0)

This contains the contents of the ELF interpreter

information passed to the process at exec time. The

format is one *unsigned long* ID plus one *unsigned long*

value for each entry. The last entry contains two zeros.

See also [getauxval(3)](https://man7.org/linux/man-pages/man3/getauxval.3.html).

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/[pid]/cgroup* (since Linux 2.6.24)

See [cgroups(7)](https://man7.org/linux/man-pages/man7/cgroups.7.html).

*/proc/[pid]/clear\_refs* (since Linux 2.6.22)

This is a write-only file, writable only by owner of the

process.

The following values may be written to the file:

1 (since Linux 2.6.22)

Reset the PG\_Referenced and ACCESSED/YOUNG bits for

all the pages associated with the process. (Before

kernel 2.6.32, writing any nonzero value to this

file had this effect.)

2 (since Linux 2.6.32)

Reset the PG\_Referenced and ACCESSED/YOUNG bits for

all anonymous pages associated with the process.

3 (since Linux 2.6.32)

Reset the PG\_Referenced and ACCESSED/YOUNG bits for

all file-mapped pages associated with the process.

Clearing the PG\_Referenced and ACCESSED/YOUNG bits

provides a method to measure approximately how much memory

a process is using. One first inspects the values in the

"Referenced" fields for the VMAs shown in

*/proc/[pid]/smaps* to get an idea of the memory footprint

of the process. One then clears the PG\_Referenced and

ACCESSED/YOUNG bits and, after some measured time

interval, once again inspects the values in the

"Referenced" fields to get an idea of the change in memory

footprint of the process during the measured interval. If

one is interested only in inspecting the selected mapping

types, then the value 2 or 3 can be used instead of 1.

Further values can be written to affect different

properties:

4 (since Linux 3.11)

Clear the soft-dirty bit for all the pages

associated with the process. This is used (in

conjunction with */proc/[pid]/pagemap*) by the check-

point restore system to discover which pages of a

process have been dirtied since the file

*/proc/[pid]/clear\_refs* was written to.

5 (since Linux 4.0)

Reset the peak resident set size ("high water

mark") to the process's current resident set size

value.

Writing any value to */proc/[pid]/clear\_refs* other than

those listed above has no effect.

The */proc/[pid]/clear\_refs* file is present only if the

**CONFIG\_PROC\_PAGE\_MONITOR** kernel configuration option is

enabled.

*/proc/[pid]/cmdline*

This read-only file holds the complete command line for

the process, unless the process is a zombie. In the

latter case, there is nothing in this file: that is, a

read on this file will return 0 characters. The command-

line arguments appear in this file as a set of strings

separated by null bytes ('\0'), with a further null byte

after the last string.

If, after an [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html), the process modifies its *argv*

strings, those changes will show up here. This is not the

same thing as modifying the *argv* array.

Furthermore, a process may change the memory location that

this file refers via [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html) operations such as

**PR\_SET\_MM\_ARG\_START**.

Think of this file as the command line that the process

wants you to see.

Этот файл только для чтения содержит полную командную строку для

процесс, если только этот процесс не является зомби. в

В последнем случае в этом файле ничего нет: т.е.

read в этом файле вернет 0 символов. Команда-

строковые аргументы отображаются в этом файле как набор строк

разделенные нулевыми байтами ('\0'), с дополнительным нулевым байтом

после последней строки.

Если после execve(2) процесс изменяет свой argv

строки, эти изменения будут отображаться здесь. Это не

то же самое, что и изменение массива argv.

Кроме того, процесс может изменить место в памяти, которое

этот файл ссылается через операции prctl(2), такие как

PR\_SET\_MM\_ARG\_START.

Думайте об этом файле как о командной строке, которую процесс

хочет, чтобы ты увидел.

*/proc/[pid]/comm* (since Linux 2.6.33)

This file exposes the process's *comm* value—that is, the

command name associated with the process. Different

threads in the same process may have different *comm*

values, accessible via */proc/[pid]/task/[tid]/comm*. A

thread may modify its *comm* value, or that of any of other

thread in the same thread group (see the discussion of

**CLONE\_THREAD** in [clone(2)](https://man7.org/linux/man-pages/man2/clone.2.html)), by writing to the file

*/proc/self/task/[tid]/comm*. Strings longer than

**TASK\_COMM\_LEN** (16) characters (including the terminating

null byte) are silently truncated.

This file provides a superset of the [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html) **PR\_SET\_NAME**

and **PR\_GET\_NAME** operations, and is employed by

[pthread\_setname\_np(3)](https://man7.org/linux/man-pages/man3/pthread_setname_np.3.html) when used to rename threads other

than the caller. The value in this file is used for the

*%e* specifier in */proc/sys/kernel/core\_pattern*; see

[core(5)](https://man7.org/linux/man-pages/man5/core.5.html).

Этот файл предоставляет значение связи процесса, т. е.

имя команды, связанное с процессом. Разные

потоки в одном и том же процессе могут иметь разные

значения, доступные через /proc/[pid]/task/[tid]/comm. А

поток может изменить свое значение связи или значение любого другого

поток в той же группе потоков (см. обсуждение

CLONE\_THREAD в clone(2)), записав в файл

/proc/self/task/[tid]/comm. Строки длиннее, чем

TASK\_COMM\_LEN (16) символов (включая завершающий

нулевой байт) автоматически усекаются.

Этот файл предоставляет надмножество prctl(2) PR\_SET\_NAME.

и PR\_GET\_NAME, и используется

pthread\_setname\_np(3) при использовании для переименования потоков других

чем вызывающий. Значение в этом файле используется для

Спецификатор %e в /proc/sys/kernel/core\_pattern; видеть

ядро (5).

*/proc/[pid]/coredump\_filter* (since Linux 2.6.23)

See [core(5)](https://man7.org/linux/man-pages/man5/core.5.html).

*/proc/[pid]/cpuset* (since Linux 2.6.12)

See [cpuset(7)](https://man7.org/linux/man-pages/man7/cpuset.7.html).

*/proc/[pid]/cwd*

This is a symbolic link to the current working directory

of the process. To find out the current working directory

of process 20, for instance, you can do this:

$ **cd /proc/20/cwd; pwd -P**

In a multithreaded process, the contents of this symbolic

link are not available if the main thread has already

terminated (typically by calling [pthread\_exit(3)](https://man7.org/linux/man-pages/man3/pthread_exit.3.html)).

Permission to dereference or read ([readlink(2)](https://man7.org/linux/man-pages/man2/readlink.2.html)) this

symbolic link is governed by a ptrace access mode

**PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/[pid]/environ*

This file contains the initial environment that was set

when the currently executing program was started via

[execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html). The entries are separated by null bytes

('\0'), and there may be a null byte at the end. Thus, to

print out the environment of process 1, you would do:

$ **cat /proc/1/environ | tr '\000' '\n'**

If, after an [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html), the process modifies its

environment (e.g., by calling functions such as [putenv(3)](https://man7.org/linux/man-pages/man3/putenv.3.html)

or modifying the [environ(7)](https://man7.org/linux/man-pages/man7/environ.7.html) variable directly), this file

will *not* reflect those changes.

Furthermore, a process may change the memory location that

this file refers via [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html) operations such as

**PR\_SET\_MM\_ENV\_START**.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/[pid]/exe*

Under Linux 2.2 and later, this file is a symbolic link containing the actual pathname of the executed command. This symbolic link can be dereferenced normally;

attempting to open it will open the executable. You can

even type */proc/[pid]/exe* to run another copy of the same

executable that is being run by process [pid]. If the

pathname has been unlinked, the symbolic link will contain

the string '(deleted)' appended to the original pathname.

In a multithreaded process, the contents of this symbolic

link are not available if the main thread has already

terminated (typically by calling [pthread\_exit(3)](https://man7.org/linux/man-pages/man3/pthread_exit.3.html)).

Permission to dereference or read ([readlink(2)](https://man7.org/linux/man-pages/man2/readlink.2.html)) this

symbolic link is governed by a ptrace access mode

**PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

Under Linux 2.0 and earlier, */proc/[pid]/exe* is a pointer

to the binary which was executed, and appears as a

symbolic link. A [readlink(2)](https://man7.org/linux/man-pages/man2/readlink.2.html) call on this file under

Linux 2.0 returns a string in the format:

[device]:inode

For example, [0301]:1502 would be inode 1502 on device

major 03 (IDE, MFM, etc. drives) minor 01 (first partition

on the first drive).

[find(1)](https://man7.org/linux/man-pages/man1/find.1.html) with the *-inum* option can be used to locate the

file.

*/proc/[pid]/fd/*

This is a subdirectory containing one entry for each file

which the process has open, named by its file descriptor,

and which is a symbolic link to the actual file. Thus, 0

is standard input, 1 standard output, 2 standard error,

and so on.

For file descriptors for pipes and sockets, the entries

will be symbolic links whose content is the file type with

the inode. A [readlink(2)](https://man7.org/linux/man-pages/man2/readlink.2.html) call on this file returns a

string in the format:

type:[inode]

For example, *socket:[2248868]* will be a socket and its

inode is 2248868. For sockets, that inode can be used to

find more information in one of the files under

*/proc/net/*.

For file descriptors that have no corresponding inode

(e.g., file descriptors produced by [bpf(2)](https://man7.org/linux/man-pages/man2/bpf.2.html),

[epoll\_create(2)](https://man7.org/linux/man-pages/man2/epoll_create.2.html), [eventfd(2)](https://man7.org/linux/man-pages/man2/eventfd.2.html), [inotify\_init(2)](https://man7.org/linux/man-pages/man2/inotify_init.2.html),

[perf\_event\_open(2)](https://man7.org/linux/man-pages/man2/perf_event_open.2.html), [signalfd(2)](https://man7.org/linux/man-pages/man2/signalfd.2.html), [timerfd\_create(2)](https://man7.org/linux/man-pages/man2/timerfd_create.2.html), and

[userfaultfd(2)](https://man7.org/linux/man-pages/man2/userfaultfd.2.html)), the entry will be a symbolic link with

contents of the form

anon\_inode:<file-type>

In many cases (but not all), the *file-type* is surrounded

by square brackets.

For example, an epoll file descriptor will have a symbolic

link whose content is the string *anon\_inode:[eventpoll]*.

In a multithreaded process, the contents of this directory

are not available if the main thread has already

terminated (typically by calling [pthread\_exit(3)](https://man7.org/linux/man-pages/man3/pthread_exit.3.html)).

Programs that take a filename as a command-line argument,

but don't take input from standard input if no argument is

supplied, and programs that write to a file named as a

command-line argument, but don't send their output to

standard output if no argument is supplied, can

nevertheless be made to use standard input or standard

output by using */proc/[pid]/fd* files as command-line

arguments. For example, assuming that *-i* is the flag

designating an input file and *-o* is the flag designating

an output file:

$ **foobar -i /proc/self/fd/0 -o /proc/self/fd/1 ...**

and you have a working filter.

*/proc/self/fd/N* is approximately the same as */dev/fd/N* in

some UNIX and UNIX-like systems. Most Linux MAKEDEV

scripts symbolically link */dev/fd* to */proc/self/fd*, in

fact.

Most systems provide symbolic links */dev/stdin*,

*/dev/stdout*, and */dev/stderr*, which respectively link to

the files *0*, *1*, and *2* in */proc/self/fd*. Thus the example

command above could be written as:

$ **foobar -i /dev/stdin -o /dev/stdout ...**

Permission to dereference or read ([readlink(2)](https://man7.org/linux/man-pages/man2/readlink.2.html)) the

symbolic links in this directory is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

Note that for file descriptors referring to inodes (pipes

and sockets, see above), those inodes still have

permission bits and ownership information distinct from

those of the */proc/[pid]/fd* entry, and that the owner may

differ from the user and group IDs of the process. An

unprivileged process may lack permissions to open them, as

in this example:

$ **echo test | sudo -u nobody cat**

test

$ **echo test | sudo -u nobody cat /proc/self/fd/0**

cat: /proc/self/fd/0: Permission denied

File descriptor 0 refers to the pipe created by the shell

and owned by that shell's user, which is not *nobody*, so

**cat** does not have permission to create a new file

descriptor to read from that inode, even though it can

still read from its existing file descriptor 0.

*/proc/[pid]/fdinfo/* (since Linux 2.6.22)

This is a subdirectory containing one entry for each file

which the process has open, named by its file descriptor.

The files in this directory are readable only by the owner

of the process. The contents of each file can be read to

obtain information about the corresponding file

descriptor. The content depends on the type of file

referred to by the corresponding file descriptor.

For regular files and directories, we see something like:

$ **cat /proc/12015/fdinfo/4**

pos: 1000

flags: 01002002

mnt\_id: 21

The fields are as follows:

*pos* This is a decimal number showing the file offset.

*flags* This is an octal number that displays the file

access mode and file status flags (see [open(2)](https://man7.org/linux/man-pages/man2/open.2.html)).

If the close-on-exec file descriptor flag is set,

then *flags* will also include the value **O\_CLOEXEC**.

Before Linux 3.1, this field incorrectly displayed

the setting of **O\_CLOEXEC** at the time the file was

opened, rather than the current setting of the

close-on-exec flag.

*mnt\_id* This field, present since Linux 3.15, is the ID of

the mount containing this file. See the

description of */proc/[pid]/mountinfo*.

For eventfd file descriptors (see [eventfd(2)](https://man7.org/linux/man-pages/man2/eventfd.2.html)), we see

(since Linux 3.8) the following fields:

pos: 0

flags: 02

mnt\_id: 10

eventfd-count: 40

*eventfd-count* is the current value of the eventfd counter,

in hexadecimal.

For epoll file descriptors (see [epoll(7)](https://man7.org/linux/man-pages/man7/epoll.7.html)), we see (since

Linux 3.8) the following fields:

pos: 0

flags: 02

mnt\_id: 10

tfd: 9 events: 19 data: 74253d2500000009

tfd: 7 events: 19 data: 74253d2500000007

Each of the lines beginning *tfd* describes one of the file

descriptors being monitored via the epoll file descriptor

(see [epoll\_ctl(2)](https://man7.org/linux/man-pages/man2/epoll_ctl.2.html) for some details). The *tfd* field is the

number of the file descriptor. The *events* field is a

hexadecimal mask of the events being monitored for this

file descriptor. The *data* field is the data value

associated with this file descriptor.

For signalfd file descriptors (see [signalfd(2)](https://man7.org/linux/man-pages/man2/signalfd.2.html)), we see

(since Linux 3.8) the following fields:

pos: 0

flags: 02

mnt\_id: 10

sigmask: 0000000000000006

*sigmask* is the hexadecimal mask of signals that are

accepted via this signalfd file descriptor. (In this

example, bits 2 and 3 are set, corresponding to the

signals **SIGINT** and **SIGQUIT**; see [signal(7)](https://man7.org/linux/man-pages/man7/signal.7.html).)

For inotify file descriptors (see [inotify(7)](https://man7.org/linux/man-pages/man7/inotify.7.html)), we see

(since Linux 3.8) the following fields:

pos: 0

flags: 00

mnt\_id: 11

inotify wd:2 ino:7ef82a sdev:800001 mask:800afff ignored\_mask:0 fhandle-bytes:8 fhandle-type:1 f\_handle:2af87e00220ffd73

inotify wd:1 ino:192627 sdev:800001 mask:800afff ignored\_mask:0 fhandle-bytes:8 fhandle-type:1 f\_handle:27261900802dfd73

Each of the lines beginning with "inotify" displays

information about one file or directory that is being

monitored. The fields in this line are as follows:

*wd* A watch descriptor number (in decimal).

*ino* The inode number of the target file (in

hexadecimal).

*sdev* The ID of the device where the target file resides

(in hexadecimal).

*mask* The mask of events being monitored for the target

file (in hexadecimal).

If the kernel was built with exportfs support, the path to

the target file is exposed as a file handle, via three

hexadecimal fields: *fhandle-bytes*, *fhandle-type*, and

*f\_handle*.

For fanotify file descriptors (see [fanotify(7)](https://man7.org/linux/man-pages/man7/fanotify.7.html)), we see

(since Linux 3.8) the following fields:

pos: 0

flags: 02

mnt\_id: 11

fanotify flags:0 event-flags:88002

fanotify ino:19264f sdev:800001 mflags:0 mask:1 ignored\_mask:0 fhandle-bytes:8 fhandle-type:1 f\_handle:4f261900a82dfd73

The fourth line displays information defined when the

fanotify group was created via [fanotify\_init(2)](https://man7.org/linux/man-pages/man2/fanotify_init.2.html):

*flags* The *flags* argument given to [fanotify\_init(2)](https://man7.org/linux/man-pages/man2/fanotify_init.2.html)

(expressed in hexadecimal).

*event-flags*

The *event\_f\_flags* argument given to

[fanotify\_init(2)](https://man7.org/linux/man-pages/man2/fanotify_init.2.html) (expressed in hexadecimal).

Each additional line shown in the file contains

information about one of the marks in the fanotify group.

Most of these fields are as for inotify, except:

*mflags* The flags associated with the mark (expressed in

hexadecimal).

*mask* The events mask for this mark (expressed in

hexadecimal).

*ignored\_mask*

The mask of events that are ignored for this mark

(expressed in hexadecimal).

For details on these fields, see [fanotify\_mark(2)](https://man7.org/linux/man-pages/man2/fanotify_mark.2.html).

For timerfd file descriptors (see **timerfd**(2)), we see

(since Linux 3.17) the following fields:

pos: 0

flags: 02004002

mnt\_id: 13

clockid: 0

ticks: 0

settime flags: 03

it\_value: (7695568592, 640020877)

it\_interval: (0, 0)

*clockid*

This is the numeric value of the clock ID

(corresponding to one of the **CLOCK\_\*** constants

defined via *<time.h>*) that is used to mark the

progress of the timer (in this example, 0 is

**CLOCK\_REALTIME**).

*ticks* This is the number of timer expirations that have

occurred, (i.e., the value that [read(2)](https://man7.org/linux/man-pages/man2/read.2.html) on it would

return).

*settime flags*

This field lists the flags with which the timerfd

was last armed (see [timerfd\_settime(2)](https://man7.org/linux/man-pages/man2/timerfd_settime.2.html)), in octal

(in this example, both **TFD\_TIMER\_ABSTIME** and

**TFD\_TIMER\_CANCEL\_ON\_SET** are set).

*it\_value*

This field contains the amount of time until the

timer will next expire, expressed in seconds and

nanoseconds. This is always expressed as a

relative value, regardless of whether the timer was

created using the **TFD\_TIMER\_ABSTIME** flag.

*it\_interval*

This field contains the interval of the timer, in

seconds and nanoseconds. (The *it\_value* and

*it\_interval* fields contain the values that

[timerfd\_gettime(2)](https://man7.org/linux/man-pages/man2/timerfd_gettime.2.html) on this file descriptor would

return.)

*/proc/[pid]/gid\_map* (since Linux 3.5)

See [user\_namespaces(7)](https://man7.org/linux/man-pages/man7/user_namespaces.7.html).

*/proc/[pid]/io* (since kernel 2.6.20)

This file contains I/O statistics for the process, for

example:

# **cat /proc/3828/io**

rchar: 323934931

wchar: 323929600

syscr: 632687

syscw: 632675

read\_bytes: 0

write\_bytes: 323932160

cancelled\_write\_bytes: 0

The fields are as follows:

*rchar*: characters read

The number of bytes which this task has caused to

be read from storage. This is simply the sum of

bytes which this process passed to [read(2)](https://man7.org/linux/man-pages/man2/read.2.html) and

similar system calls. It includes things such as

terminal I/O and is unaffected by whether or not

actual physical disk I/O was required (the read

might have been satisfied from pagecache).

*wchar*: characters written

The number of bytes which this task has caused, or

shall cause to be written to disk. Similar caveats

apply here as with *rchar*.

*syscr*: read syscalls

Attempt to count the number of read I/O operations—

that is, system calls such as [read(2)](https://man7.org/linux/man-pages/man2/read.2.html) and [pread(2)](https://man7.org/linux/man-pages/man2/pread.2.html).

*syscw*: write syscalls

Attempt to count the number of write I/O

operations—that is, system calls such as [write(2)](https://man7.org/linux/man-pages/man2/write.2.html)

and [pwrite(2)](https://man7.org/linux/man-pages/man2/pwrite.2.html).

*read\_bytes*: bytes read

Attempt to count the number of bytes which this

process really did cause to be fetched from the

storage layer. This is accurate for block-backed

filesystems.

*write\_bytes*: bytes written

Attempt to count the number of bytes which this

process caused to be sent to the storage layer.

*cancelled\_write\_bytes*:

The big inaccuracy here is truncate. If a process

writes 1 MB to a file and then deletes the file, it

will in fact perform no writeout. But it will have

been accounted as having caused 1 MB of write. In

other words: this field represents the number of

bytes which this process caused to not happen, by

truncating pagecache. A task can cause "negative"

I/O too. If this task truncates some dirty

pagecache, some I/O which another task has been

accounted for (in its *write\_bytes*) will not be

happening.

*Note*: In the current implementation, things are a bit racy

on 32-bit systems: if process A reads process B's

*/proc/[pid]/io* while process B is updating one of these

64-bit counters, process A could see an intermediate

result.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/[pid]/limits* (since Linux 2.6.24)

This file displays the soft limit, hard limit, and units

of measurement for each of the process's resource limits

(see [getrlimit(2)](https://man7.org/linux/man-pages/man2/getrlimit.2.html)). Up to and including Linux 2.6.35,

this file is protected to allow reading only by the real

UID of the process. Since Linux 2.6.36, this file is

readable by all users on the system.

*/proc/[pid]/map\_files/* (since kernel 3.3)

This subdirectory contains entries corresponding to

memory-mapped files (see [mmap(2)](https://man7.org/linux/man-pages/man2/mmap.2.html)). Entries are named by

memory region start and end address pair (expressed as

hexadecimal numbers), and are symbolic links to the mapped

files themselves. Here is an example, with the output

wrapped and reformatted to fit on an 80-column display:

# **ls -l /proc/self/map\_files/**

lr--------. 1 root root 64 Apr 16 21:31

3252e00000-3252e20000 -> /usr/lib64/ld-2.15.so

...

Although these entries are present for memory regions that

were mapped with the **MAP\_FILE** flag, the way anonymous

shared memory (regions created with the **MAP\_ANON |**

**MAP\_SHARED** flags) is implemented in Linux means that such

regions also appear on this directory. Here is an example

where the target file is the deleted */dev/zero* one:

lrw-------. 1 root root 64 Apr 16 21:33

7fc075d2f000-7fc075e6f000 -> /dev/zero (deleted)

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

Until kernel version 4.3, this directory appeared only if

the **CONFIG\_CHECKPOINT\_RESTORE** kernel configuration option

was enabled.

Capabilities are required to read the contents of the

symbolic links in this directory: before Linux 5.9, the

reading process requires **CAP\_SYS\_ADMIN** in the initial user

namespace; since Linux 5.9, the reading process must have

either **CAP\_SYS\_ADMIN** or **CAP\_CHECKPOINT\_RESTORE** in the user

namespace where it resides.

***/proc/[pid]/maps***

A file containing the currently mapped memory regions and

their access permissions. See [mmap(2)](https://man7.org/linux/man-pages/man2/mmap.2.html) for some further

information about memory mappings.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

The format of the file is:

*address* *perms offset* *dev* *inode* *pathname*

00400000-00452000 r-xp 00000000 08:02 173521 /usr/bin/dbus-daemon

00651000-00652000 r--p 00051000 08:02 173521 /usr/bin/dbus-daemon

00652000-00655000 rw-p 00052000 08:02 173521 /usr/bin/dbus-daemon

00e03000-00e24000 rw-p 00000000 00:00 0 [heap]

00e24000-011f7000 rw-p 00000000 00:00 0 [heap]

...

35b1800000-35b1820000 r-xp 00000000 08:02 135522 /usr/lib64/ld-2.15.so

35b1a1f000-35b1a20000 r--p 0001f000 08:02 135522 /usr/lib64/ld-2.15.so

35b1a20000-35b1a21000 rw-p 00020000 08:02 135522 /usr/lib64/ld-2.15.so

35b1a21000-35b1a22000 rw-p 00000000 00:00 0

35b1c00000-35b1dac000 r-xp 00000000 08:02 135870 /usr/lib64/libc-2.15.so

35b1dac000-35b1fac000 ---p 001ac000 08:02 135870 /usr/lib64/libc-2.15.so

35b1fac000-35b1fb0000 r--p 001ac000 08:02 135870 /usr/lib64/libc-2.15.so

35b1fb0000-35b1fb2000 rw-p 001b0000 08:02 135870 /usr/lib64/libc-2.15.so

...

f2c6ff8c000-7f2c7078c000 rw-p 00000000 00:00 0 [stack:986]

...

7fffb2c0d000-7fffb2c2e000 rw-p 00000000 00:00 0 [stack]

7fffb2d48000-7fffb2d49000 r-xp 00000000 00:00 0 [vdso]

The *address* field is the address space in the process that

the mapping occupies. The *perms* field is a set of

permissions:

r = read

w = write

x = execute

s = shared

p = private (copy on write)

The *offset* field is the offset into the file/whatever; *dev*

is the device (major:minor); *inode* is the inode on that

device. 0 indicates that no inode is associated with the

memory region, as would be the case with BSS

(uninitialized data).

The *pathname* field will usually be the file that is

backing the mapping. For ELF files, you can easily

coordinate with the *offset* field by looking at the Offset

field in the ELF program headers (*readelf -l*).

There are additional helpful pseudo-paths:

*[stack]*

The initial process's (also known as the main

thread's) stack.

*[stack:<tid>]* (from Linux 3.4 to 4.4)

A thread's stack (where the *<tid>* is a thread ID).

It corresponds to the */proc/[pid]/task/[tid]/* path.

This field was removed in Linux 4.5, since

providing this information for a process with large

numbers of threads is expensive.

*[vdso]* The virtual dynamically linked shared object. See

[vdso(7)](https://man7.org/linux/man-pages/man7/vdso.7.html).

*[heap]* The process's heap.

If the *pathname* field is blank, this is an anonymous

mapping as obtained via [mmap(2)](https://man7.org/linux/man-pages/man2/mmap.2.html). There is no easy way to

coordinate this back to a process's source, short of

running it through [gdb(1)](https://man7.org/linux/man-pages/man1/gdb.1.html), [strace(1)](https://man7.org/linux/man-pages/man1/strace.1.html), or similar.

*pathname* is shown unescaped except for newline characters,

which are replaced with an octal escape sequence. As a

result, it is not possible to determine whether the

original pathname contained a newline character or the

literal *\012* character sequence.

If the mapping is file-backed and the file has been

deleted, the string " (deleted)" is appended to the

pathname. Note that this is ambiguous too.

Under Linux 2.0, there is no field giving pathname.

*/proc/[pid]/mem*

This file can be used to access the pages of a process's

memory through [open(2)](https://man7.org/linux/man-pages/man2/open.2.html), [read(2)](https://man7.org/linux/man-pages/man2/read.2.html), and [lseek(2)](https://man7.org/linux/man-pages/man2/lseek.2.html).

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_ATTACH\_FSCREDS** check; see

[ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

***/proc/[pid]/mountinfo* (since Linux 2.6.26)**

This file contains information about mounts in the

process's mount namespace (see [mount\_namespaces(7)](https://man7.org/linux/man-pages/man7/mount_namespaces.7.html)). It

supplies various information (e.g., propagation state,

root of mount for bind mounts, identifier for each mount

and its parent) that is missing from the (older)

*/proc/[pid]/mounts* file, and fixes various other problems

with that file (e.g., nonextensibility, failure to

distinguish per-mount versus per-superblock options).

The file contains lines of the form:

36 35 98:0 /mnt1 /mnt2 rw,noatime master:1 - ext3 /dev/root rw,errors=continue

(1)(2)(3) (4) (5) (6) (7) (8) (9) (10) (11)

The numbers in parentheses are labels for the descriptions

below:

(1) mount ID: a unique ID for the mount (may be reused

after [umount(2)](https://man7.org/linux/man-pages/man2/umount.2.html)).

(2) parent ID: the ID of the parent mount (or of self for

the root of this mount namespace's mount tree).

If a new mount is stacked on top of a previous

existing mount (so that it hides the existing mount)

at pathname P, then the parent of the new mount is

the previous mount at that location. Thus, when

looking at all the mounts stacked at a particular

location, the top-most mount is the one that is not

the parent of any other mount at the same location.

(Note, however, that this top-most mount will be

accessible only if the longest path subprefix of P

that is a mount point is not itself hidden by a

stacked mount.)

If the parent mount lies outside the process's root

directory (see [chroot(2)](https://man7.org/linux/man-pages/man2/chroot.2.html)), the ID shown here won't

have a corresponding record in *mountinfo* whose mount

ID (field 1) matches this parent mount ID (because

mounts that lie outside the process's root directory

are not shown in *mountinfo*). As a special case of

this point, the process's root mount may have a

parent mount (for the initramfs filesystem) that lies

outside the process's root directory, and an entry

for that mount will not appear in *mountinfo*.

(3) major:minor: the value of *st\_dev* for files on this

filesystem (see [stat(2)](https://man7.org/linux/man-pages/man2/stat.2.html)).

(4) root: the pathname of the directory in the filesystem

which forms the root of this mount.

(5) mount point: the pathname of the mount point relative

to the process's root directory.

(6) mount options: per-mount options (see [mount(2)](https://man7.org/linux/man-pages/man2/mount.2.html)).

(7) optional fields: zero or more fields of the form

"tag[:value]"; see below.

(8) separator: the end of the optional fields is marked

by a single hyphen.

(9) filesystem type: the filesystem type in the form

"type[.subtype]".

(10) mount source: filesystem-specific information or

"none".

(11) super options: per-superblock options (see [mount(2)](https://man7.org/linux/man-pages/man2/mount.2.html)).

Currently, the possible optional fields are *shared*,

*master*, *propagate\_from*, and *unbindable*. See

[mount\_namespaces(7)](https://man7.org/linux/man-pages/man7/mount_namespaces.7.html) for a description of these fields.

Parsers should ignore all unrecognized optional fields.

For more information on mount propagation see:

*Documentation/filesystems/sharedsubtree.txt* in the Linux

kernel source tree.

*/proc/[pid]/mounts* (since Linux 2.4.19)

This file lists all the filesystems currently mounted in

the process's mount namespace (see [mount\_namespaces(7)](https://man7.org/linux/man-pages/man7/mount_namespaces.7.html)).

The format of this file is documented in [fstab(5)](https://man7.org/linux/man-pages/man5/fstab.5.html).

Since kernel version 2.6.15, this file is pollable: after

opening the file for reading, a change in this file (i.e.,

a filesystem mount or unmount) causes [select(2)](https://man7.org/linux/man-pages/man2/select.2.html) to mark

the file descriptor as having an exceptional condition,

and [poll(2)](https://man7.org/linux/man-pages/man2/poll.2.html) and [epoll\_wait(2)](https://man7.org/linux/man-pages/man2/epoll_wait.2.html) mark the file as having a

priority event (**POLLPRI**). (Before Linux 2.6.30, a change

in this file was indicated by the file descriptor being

marked as readable for [select(2)](https://man7.org/linux/man-pages/man2/select.2.html), and being marked as

having an error condition for [poll(2)](https://man7.org/linux/man-pages/man2/poll.2.html) and [epoll\_wait(2)](https://man7.org/linux/man-pages/man2/epoll_wait.2.html).)

*/proc/[pid]/mountstats* (since Linux 2.6.17)

This file exports information (statistics, configuration

information) about the mounts in the process's mount

namespace (see [mount\_namespaces(7)](https://man7.org/linux/man-pages/man7/mount_namespaces.7.html)). Lines in this file

have the form:

device /dev/sda7 mounted on /home with fstype ext3 [stats]

( 1 ) ( 2 ) (3 ) ( 4 )

The fields in each line are:

(1) The name of the mounted device (or "nodevice" if

there is no corresponding device).

(2) The mount point within the filesystem tree.

(3) The filesystem type.

(4) Optional statistics and configuration information.

Currently (as at Linux 2.6.26), only NFS filesystems

export information via this field.

This file is readable only by the owner of the process.

*/proc/[pid]/net* (since Linux 2.6.25)

See the description of */proc/net*.

*/proc/[pid]/ns/* (since Linux 3.0)

This is a subdirectory containing one entry for each

namespace that supports being manipulated by [setns(2)](https://man7.org/linux/man-pages/man2/setns.2.html).

For more information, see [namespaces(7)](https://man7.org/linux/man-pages/man7/namespaces.7.html).

*/proc/[pid]/numa\_maps* (since Linux 2.6.14)

See [numa(7)](https://man7.org/linux/man-pages/man7/numa.7.html).

*/proc/[pid]/oom\_adj* (since Linux 2.6.11)

This file can be used to adjust the score used to select

which process should be killed in an out-of-memory (OOM)

situation. The kernel uses this value for a bit-shift

operation of the process's *oom\_score* value: valid values

are in the range -16 to +15, plus the special value -17,

which disables OOM-killing altogether for this process. A

positive score increases the likelihood of this process

being killed by the OOM-killer; a negative score decreases

the likelihood.

The default value for this file is 0; a new process

inherits its parent's *oom\_adj* setting. A process must be

privileged (**CAP\_SYS\_RESOURCE**) to update this file.

Since Linux 2.6.36, use of this file is deprecated in

favor of */proc/[pid]/oom\_score\_adj*.

*/proc/[pid]/oom\_score* (since Linux 2.6.11)

This file displays the current score that the kernel gives

to this process for the purpose of selecting a process for

the OOM-killer. A higher score means that the process is

more likely to be selected by the OOM-killer. The basis

for this score is the amount of memory used by the

process, with increases (+) or decreases (-) for factors

including:

\* whether the process is privileged (-).

Before kernel 2.6.36 the following factors were also used

in the calculation of oom\_score:

\* whether the process creates a lot of children using

[fork(2)](https://man7.org/linux/man-pages/man2/fork.2.html) (+);

\* whether the process has been running a long time, or has

used a lot of CPU time (-);

\* whether the process has a low nice value (i.e., > 0)

(+); and

\* whether the process is making direct hardware access

(-).

The *oom\_score* also reflects the adjustment specified by

the *oom\_score\_adj* or *oom\_adj* setting for the process.

*/proc/[pid]/oom\_score\_adj* (since Linux 2.6.36)

This file can be used to adjust the badness heuristic used

to select which process gets killed in out-of-memory

conditions.

The badness heuristic assigns a value to each candidate

task ranging from 0 (never kill) to 1000 (always kill) to

determine which process is targeted. The units are

roughly a proportion along that range of allowed memory

the process may allocate from, based on an estimation of

its current memory and swap use. For example, if a task

is using all allowed memory, its badness score will be

1000. If it is using half of its allowed memory, its

score will be 500.

There is an additional factor included in the badness

score: root processes are given 3% extra memory over other

tasks.

The amount of "allowed" memory depends on the context in

which the OOM-killer was called. If it is due to the

memory assigned to the allocating task's cpuset being

exhausted, the allowed memory represents the set of mems

assigned to that cpuset (see [cpuset(7)](https://man7.org/linux/man-pages/man7/cpuset.7.html)). If it is due to

a mempolicy's node(s) being exhausted, the allowed memory

represents the set of mempolicy nodes. If it is due to a

memory limit (or swap limit) being reached, the allowed

memory is that configured limit. Finally, if it is due to

the entire system being out of memory, the allowed memory

represents all allocatable resources.

The value of *oom\_score\_adj* is added to the badness score

before it is used to determine which task to kill.

Acceptable values range from -1000 (OOM\_SCORE\_ADJ\_MIN) to

+1000 (OOM\_SCORE\_ADJ\_MAX). This allows user space to

control the preference for OOM-killing, ranging from

always preferring a certain task or completely disabling

it from OOM-killing. The lowest possible value, -1000, is

equivalent to disabling OOM-killing entirely for that

task, since it will always report a badness score of 0.

Consequently, it is very simple for user space to define

the amount of memory to consider for each task. Setting

an *oom\_score\_adj* value of +500, for example, is roughly

equivalent to allowing the remainder of tasks sharing the

same system, cpuset, mempolicy, or memory controller

resources to use at least 50% more memory. A value of

-500, on the other hand, would be roughly equivalent to

discounting 50% of the task's allowed memory from being

considered as scoring against the task.

For backward compatibility with previous kernels,

*/proc/[pid]/oom\_adj* can still be used to tune the badness

score. Its value is scaled linearly with *oom\_score\_adj*.

Writing to */proc/[pid]/oom\_score\_adj* or

*/proc/[pid]/oom\_adj* will change the other with its scaled

value.

The [choom(1)](https://man7.org/linux/man-pages/man1/choom.1.html) program provides a command-line interface for

adjusting the *oom\_score\_adj* value of a running process or

a newly executed command.

***/proc/[pid]/pagemap* (since Linux 2.6.25)**

This file shows the mapping of each of the process's

virtual pages into physical page frames or swap area. It

contains one 64-bit value for each virtual page, with the

bits set as follows:

63 If set, the page is present in RAM.

62 If set, the page is in swap space

61 (since Linux 3.5)

The page is a file-mapped page or a shared

anonymous page.

60–57 (since Linux 3.11)

Zero

56 (since Linux 4.2)

The page is exclusively mapped.

55 (since Linux 3.11)

PTE is soft-dirty (see the kernel source file

*Documentation/admin-guide/mm/soft-dirty.rst*).

54–0 If the page is present in RAM (bit 63), then these

bits provide the page frame number, which can be

used to index */proc/kpageflags* and

*/proc/kpagecount*. If the page is present in swap

(bit 62), then bits 4–0 give the swap type, and

bits 54–5 encode the swap offset.

Before Linux 3.11, bits 60–55 were used to encode the

base-2 log of the page size.

To employ */proc/[pid]/pagemap* efficiently, use

*/proc/[pid]/maps* to determine which areas of memory are

actually mapped and seek to skip over unmapped regions.

The */proc/[pid]/pagemap* file is present only if the

**CONFIG\_PROC\_PAGE\_MONITOR** kernel configuration option is

enabled.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/[pid]/personality* (since Linux 2.6.28)

This read-only file exposes the process's execution

domain, as set by [personality(2)](https://man7.org/linux/man-pages/man2/personality.2.html). The value is displayed

in hexadecimal notation.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_ATTACH\_FSCREDS** check; see

[ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

***/proc/[pid]/root***

UNIX and Linux support the idea of a per-process root of

the filesystem, set by the [chroot(2)](https://man7.org/linux/man-pages/man2/chroot.2.html) system call. This

file is a symbolic link that points to the process's root

directory, and behaves in the same way as *exe*, and *fd/\**.

Note however that this file is not merely a symbolic link.

It provides the same view of the filesystem (including

namespaces and the set of per-process mounts) as the

process itself. An example illustrates this point. In

one terminal, we start a shell in new user and mount

namespaces, and in that shell we create some new mounts:

$ **PS1='sh1# ' unshare -Urnm**

sh1# **mount -t tmpfs tmpfs /etc** # Mount empty tmpfs at /etc

sh1# **mount --bind /usr /dev** # Mount /usr at /dev

sh1# **echo $$**

27123

In a second terminal window, in the initial mount

namespace, we look at the contents of the corresponding

mounts in the initial and new namespaces:

$ **PS1='sh2# ' sudo sh**

sh2# **ls /etc | wc -l** # In initial NS

309

sh2# **ls /proc/27123/root/etc | wc -l** # /etc in other NS

0 # The empty tmpfs dir

sh2# **ls /dev | wc -l** # In initial NS

205

sh2# **ls /proc/27123/root/dev | wc -l** # /dev in other NS

11 # Actually bind

# mounted to /usr

sh2# **ls /usr | wc -l** # /usr in initial NS

11

In a multithreaded process, the contents of the

*/proc/[pid]/root* symbolic link are not available if the

main thread has already terminated (typically by calling

[pthread\_exit(3)](https://man7.org/linux/man-pages/man3/pthread_exit.3.html)).

Permission to dereference or read ([readlink(2)](https://man7.org/linux/man-pages/man2/readlink.2.html)) this

symbolic link is governed by a ptrace access mode

**PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/[pid]/projid\_map* (since Linux 3.7)

See [user\_namespaces(7)](https://man7.org/linux/man-pages/man7/user_namespaces.7.html).

*/proc/[pid]/seccomp* (Linux 2.6.12 to 2.6.22)

This file can be used to read and change the process's

secure computing (seccomp) mode setting. It contains the

value 0 if the process is not in seccomp mode, and 1 if

the process is in strict seccomp mode (see [seccomp(2)](https://man7.org/linux/man-pages/man2/seccomp.2.html)).

Writing 1 to this file places the process irreversibly in

strict seccomp mode. (Further attempts to write to the

file fail with the **EPERM** error.)

In Linux 2.6.23, this file went away, to be replaced by

the [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html) **PR\_GET\_SECCOMP** and **PR\_SET\_SECCOMP** operations

(and later by [seccomp(2)](https://man7.org/linux/man-pages/man2/seccomp.2.html) and the *Seccomp* field in

*/proc/[pid]/status*).

*/proc/[pid]/setgroups* (since Linux 3.19)

See [user\_namespaces(7)](https://man7.org/linux/man-pages/man7/user_namespaces.7.html).

*/proc/[pid]/smaps* (since Linux 2.6.14)

This file shows memory consumption for each of the

process's mappings. (The [pmap(1)](https://man7.org/linux/man-pages/man1/pmap.1.html) command displays similar

information, in a form that may be easier for parsing.)

For each mapping there is a series of lines such as the

following:

00400000-0048a000 r-xp 00000000 fd:03 960637 /bin/bash

Size: 552 kB

Rss: 460 kB

Pss: 100 kB

Shared\_Clean: 452 kB

Shared\_Dirty: 0 kB

Private\_Clean: 8 kB

Private\_Dirty: 0 kB

Referenced: 460 kB

Anonymous: 0 kB

AnonHugePages: 0 kB

ShmemHugePages: 0 kB

ShmemPmdMapped: 0 kB

Swap: 0 kB

KernelPageSize: 4 kB

MMUPageSize: 4 kB

KernelPageSize: 4 kB

MMUPageSize: 4 kB

Locked: 0 kB

ProtectionKey: 0

VmFlags: rd ex mr mw me dw

The first of these lines shows the same information as is

displayed for the mapping in */proc/[pid]/maps*. The

following lines show the size of the mapping, the amount

of the mapping that is currently resident in RAM ("Rss"),

the process's proportional share of this mapping ("Pss"),

the number of clean and dirty shared pages in the mapping,

and the number of clean and dirty private pages in the

mapping. "Referenced" indicates the amount of memory

currently marked as referenced or accessed. "Anonymous"

shows the amount of memory that does not belong to any

file. "Swap" shows how much would-be-anonymous memory is

also used, but out on swap.

The "KernelPageSize" line (available since Linux 2.6.29)

is the page size used by the kernel to back the virtual

memory area. This matches the size used by the MMU in the

majority of cases. However, one counter-example occurs on

PPC64 kernels whereby a kernel using 64 kB as a base page

size may still use 4 kB pages for the MMU on older

processors. To distinguish the two attributes, the

"MMUPageSize" line (also available since Linux 2.6.29)

reports the page size used by the MMU.

The "Locked" indicates whether the mapping is locked in

memory or not.

The "ProtectionKey" line (available since Linux 4.9, on

x86 only) contains the memory protection key (see

[pkeys(7)](https://man7.org/linux/man-pages/man7/pkeys.7.html)) associated with the virtual memory area. This

entry is present only if the kernel was built with the

**CONFIG\_X86\_INTEL\_MEMORY\_PROTECTION\_KEYS** configuration

option (since Linux 4.6).

The "VmFlags" line (available since Linux 3.8) represents

the kernel flags associated with the virtual memory area,

encoded using the following two-letter codes:

rd - readable

wr - writable

ex - executable

sh - shared

mr - may read

mw - may write

me - may execute

ms - may share

gd - stack segment grows down

pf - pure PFN range

dw - disabled write to the mapped file

lo - pages are locked in memory

io - memory mapped I/O area

sr - sequential read advise provided

rr - random read advise provided

dc - do not copy area on fork

de - do not expand area on remapping

ac - area is accountable

nr - swap space is not reserved for the area

ht - area uses huge tlb pages

sf - perform synchronous page faults (since Linux

4.15)

nl - non-linear mapping (removed in Linux 4.0)

ar - architecture specific flag

wf - wipe on fork (since Linux 4.14)

dd - do not include area into core dump

sd - soft-dirty flag (since Linux 3.13)

mm - mixed map area

hg - huge page advise flag

nh - no-huge page advise flag

mg - mergeable advise flag

um - userfaultfd missing pages tracking (since Linux

4.3)

uw - userfaultfd wprotect pages tracking (since Linux

4.3)

The */proc/[pid]/smaps* file is present only if the

**CONFIG\_PROC\_PAGE\_MONITOR** kernel configuration option is

enabled.

*/proc/[pid]/stack* (since Linux 2.6.29)

This file provides a symbolic trace of the function calls

in this process's kernel stack. This file is provided

only if the kernel was built with the **CONFIG\_STACKTRACE**

configuration option.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_ATTACH\_FSCREDS** check; see

[ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/[pid]/stat*

Status information about the process. This is used by

[ps(1)](https://man7.org/linux/man-pages/man1/ps.1.html). It is defined in the kernel source file

*fs/proc/array.c*.

The fields, in order, with their proper [scanf(3)](https://man7.org/linux/man-pages/man3/scanf.3.html) format

specifiers, are listed below. Whether or not certain of

these fields display valid information is governed by a

ptrace access mode **PTRACE\_MODE\_READ\_FSCREDS** |

**PTRACE\_MODE\_NOAUDIT** check (refer to [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html)). If the

check denies access, then the field value is displayed as

0. The affected fields are indicated with the marking

[PT].

(1) *pid* %d

The process ID.

(2) *comm* %s

The filename of the executable, in parentheses.

Strings longer than **TASK\_COMM\_LEN** (16) characters

(including the terminating null byte) are silently

truncated. This is visible whether or not the

executable is swapped out.

(3) *state* %c

One of the following characters, indicating process

state:

R Running

S Sleeping in an interruptible wait

D Waiting in uninterruptible disk sleep

Z Zombie

T Stopped (on a signal) or (before Linux 2.6.33)

trace stopped

t Tracing stop (Linux 2.6.33 onward)

W Paging (only before Linux 2.6.0)

X Dead (from Linux 2.6.0 onward)

x Dead (Linux 2.6.33 to 3.13 only)

K Wakekill (Linux 2.6.33 to 3.13 only)

W Waking (Linux 2.6.33 to 3.13 only)

P Parked (Linux 3.9 to 3.13 only)

(4) *ppid* %d

The PID of the parent of this process.

(5) *pgrp* %d

The process group ID of the process.

(6) *session* %d

The session ID of the process.

(7) *tty\_nr* %d

The controlling terminal of the process. (The

minor device number is contained in the combination

of bits 31 to 20 and 7 to 0; the major device

number is in bits 15 to 8.)

(8) *tpgid* %d

The ID of the foreground process group of the

controlling terminal of the process.

(9) *flags* %u

The kernel flags word of the process. For bit

meanings, see the PF\_\* defines in the Linux kernel

source file *include/linux/sched.h*. Details depend

on the kernel version.

The format for this field was %lu before Linux 2.6.

(10) *minflt* %lu

The number of minor faults the process has made

which have not required loading a memory page from

disk.

(11) *cminflt* %lu

The number of minor faults that the process's

waited-for children have made.

(12) *majflt* %lu

The number of major faults the process has made

which have required loading a memory page from

disk.

(13) *cmajflt* %lu

The number of major faults that the process's

waited-for children have made.

(14) *utime* %lu

Amount of time that this process has been scheduled

in user mode, measured in clock ticks (divide by

*sysconf(\_SC\_CLK\_TCK)*). This includes guest time,

*guest\_time* (time spent running a virtual CPU, see

below), so that applications that are not aware of

the guest time field do not lose that time from

their calculations.

(15) *stime* %lu

Amount of time that this process has been scheduled

in kernel mode, measured in clock ticks (divide by

*sysconf(\_SC\_CLK\_TCK)*).

(16) *cutime* %ld

Amount of time that this process's waited-for

children have been scheduled in user mode, measured

in clock ticks (divide by *sysconf(\_SC\_CLK\_TCK)*).

(See also [times(2)](https://man7.org/linux/man-pages/man2/times.2.html).) This includes guest time,

*cguest\_time* (time spent running a virtual CPU, see

below).

(17) *cstime* %ld

Amount of time that this process's waited-for

children have been scheduled in kernel mode,

measured in clock ticks (divide by

*sysconf(\_SC\_CLK\_TCK)*).

(18) *priority* %ld

(Explanation for Linux 2.6) For processes running a

real-time scheduling policy (*policy* below; see

[sched\_setscheduler(2)](https://man7.org/linux/man-pages/man2/sched_setscheduler.2.html)), this is the negated

scheduling priority, minus one; that is, a number

in the range -2 to -100, corresponding to real-time

priorities 1 to 99. For processes running under a

non-real-time scheduling policy, this is the raw

nice value ([setpriority(2)](https://man7.org/linux/man-pages/man2/setpriority.2.html)) as represented in the

kernel. The kernel stores nice values as numbers

in the range 0 (high) to 39 (low), corresponding to

the user-visible nice range of -20 to 19.

Before Linux 2.6, this was a scaled value based on

the scheduler weighting given to this process.

(19) *nice* %ld

The nice value (see [setpriority(2)](https://man7.org/linux/man-pages/man2/setpriority.2.html)), a value in the

range 19 (low priority) to -20 (high priority).

(20) *num\_threads* %ld

Number of threads in this process (since Linux

2.6). Before kernel 2.6, this field was hard coded

to 0 as a placeholder for an earlier removed field.

(21) *itrealvalue* %ld

The time in jiffies before the next **SIGALRM** is sent

to the process due to an interval timer. Since

kernel 2.6.17, this field is no longer maintained,

and is hard coded as 0.

(22) *starttime* %llu

The time the process started after system boot. In

kernels before Linux 2.6, this value was expressed

in jiffies. Since Linux 2.6, the value is

expressed in clock ticks (divide by

*sysconf(\_SC\_CLK\_TCK)*).

The format for this field was %lu before Linux 2.6.

(23) *vsize* %lu

Virtual memory size in bytes.

(24) *rss* %ld

Resident Set Size: number of pages the process has

in real memory. This is just the pages which count

toward text, data, or stack space. This does not

include pages which have not been demand-loaded in,

or which are swapped out. This value is

inaccurate; see */proc/[pid]/statm* below.

(25) *rsslim* %lu

Current soft limit in bytes on the rss of the

process; see the description of **RLIMIT\_RSS** in

[getrlimit(2)](https://man7.org/linux/man-pages/man2/getrlimit.2.html).

(26) *startcode* %lu [PT]

The address above which program text can run.

(27) *endcode* %lu [PT]

The address below which program text can run.

(28) *startstack* %lu [PT]

The address of the start (i.e., bottom) of the

stack.

(29) *kstkesp* %lu [PT]

The current value of ESP (stack pointer), as found

in the kernel stack page for the process.

(30) *kstkeip* %lu [PT]

The current EIP (instruction pointer).

(31) *signal* %lu

The bitmap of pending signals, displayed as a

decimal number. Obsolete, because it does not

provide information on real-time signals; use

*/proc/[pid]/status* instead.

(32) *blocked* %lu

The bitmap of blocked signals, displayed as a

decimal number. Obsolete, because it does not

provide information on real-time signals; use

*/proc/[pid]/status* instead.

(33) *sigignore* %lu

The bitmap of ignored signals, displayed as a

decimal number. Obsolete, because it does not

provide information on real-time signals; use

*/proc/[pid]/status* instead.

(34) *sigcatch* %lu

The bitmap of caught signals, displayed as a

decimal number. Obsolete, because it does not

provide information on real-time signals; use

*/proc/[pid]/status* instead.

(35) *wchan* %lu [PT]

This is the "channel" in which the process is

waiting. It is the address of a location in the

kernel where the process is sleeping. The

corresponding symbolic name can be found in

*/proc/[pid]/wchan*.

(36) *nswap* %lu

Number of pages swapped (not maintained).

(37) *cnswap* %lu

Cumulative *nswap* for child processes (not

maintained).

(38) *exit\_signal* %d (since Linux 2.1.22)

Signal to be sent to parent when we die.

(39) *processor* %d (since Linux 2.2.8)

CPU number last executed on.

(40) *rt\_priority* %u (since Linux 2.5.19)

Real-time scheduling priority, a number in the

range 1 to 99 for processes scheduled under a real-

time policy, or 0, for non-real-time processes (see

[sched\_setscheduler(2)](https://man7.org/linux/man-pages/man2/sched_setscheduler.2.html)).

(41) *policy* %u (since Linux 2.5.19)

Scheduling policy (see [sched\_setscheduler(2)](https://man7.org/linux/man-pages/man2/sched_setscheduler.2.html)).

Decode using the SCHED\_\* constants in

*linux/sched.h*.

The format for this field was %lu before Linux

2.6.22.

(42) *delayacct\_blkio\_ticks* %llu (since Linux 2.6.18)

Aggregated block I/O delays, measured in clock

ticks (centiseconds).

(43) *guest\_time* %lu (since Linux 2.6.24)

Guest time of the process (time spent running a

virtual CPU for a guest operating system), measured

in clock ticks (divide by *sysconf(\_SC\_CLK\_TCK)*).

(44) *cguest\_time* %ld (since Linux 2.6.24)

Guest time of the process's children, measured in

clock ticks (divide by *sysconf(\_SC\_CLK\_TCK)*).

(45) *start\_data* %lu (since Linux 3.3) [PT]

Address above which program initialized and

uninitialized (BSS) data are placed.

(46) *end\_data* %lu (since Linux 3.3) [PT]

Address below which program initialized and

uninitialized (BSS) data are placed.

(47) *start\_brk* %lu (since Linux 3.3) [PT]

Address above which program heap can be expanded

with [brk(2)](https://man7.org/linux/man-pages/man2/brk.2.html).

(48) *arg\_start* %lu (since Linux 3.5) [PT]

Address above which program command-line arguments

(*argv*) are placed.

(49) *arg\_end* %lu (since Linux 3.5) [PT]

Address below program command-line arguments (*argv*)

are placed.

(50) *env\_start* %lu (since Linux 3.5) [PT]

Address above which program environment is placed.

(51) *env\_end* %lu (since Linux 3.5) [PT]

Address below which program environment is placed.

(52) *exit\_code* %d (since Linux 3.5) [PT]

The thread's exit status in the form reported by

[waitpid(2)](https://man7.org/linux/man-pages/man2/waitpid.2.html).

*/proc/[pid]/statm*

Provides information about memory usage, measured in

pages. The columns are:

size (1) total program size

(same as VmSize in */proc/[pid]/status*)

resident (2) resident set size

(inaccurate; same as VmRSS in */proc/[pid]/status*)

shared (3) number of resident shared pages

(i.e., backed by a file)

(inaccurate; same as RssFile+RssShmem in

*/proc/[pid]/status*)

text (4) text (code)

lib (5) library (unused since Linux 2.6; always 0)

data (6) data + stack

dt (7) dirty pages (unused since Linux 2.6; always 0)

Some of these values are inaccurate because of a kernel-

internal scalability optimization. If accurate values are

required, use */proc/[pid]/smaps* or

*/proc/[pid]/smaps\_rollup* instead, which are much slower

but provide accurate, detailed information.

***/proc/[pid]/status***

Provides much of the information in */proc/[pid]/stat* and

*/proc/[pid]/statm* in a format that's easier for humans to

parse. Here's an example:

$ **cat /proc/$$/status**

Name: bash

Umask: 0022

State: S (sleeping)

Tgid: 17248

Ngid: 0

Pid: 17248

PPid: 17200

TracerPid: 0

Uid: 1000 1000 1000 1000

Gid: 100 100 100 100

FDSize: 256

Groups: 16 33 100

NStgid: 17248

NSpid: 17248

NSpgid: 17248

NSsid: 17200

VmPeak: 131168 kB

VmSize: 131168 kB

VmLck: 0 kB

VmPin: 0 kB

VmHWM: 13484 kB

VmRSS: 13484 kB

RssAnon: 10264 kB

RssFile: 3220 kB

RssShmem: 0 kB

VmData: 10332 kB

VmStk: 136 kB

VmExe: 992 kB

VmLib: 2104 kB

VmPTE: 76 kB

VmPMD: 12 kB

VmSwap: 0 kB

HugetlbPages: 0 kB # 4.4

CoreDumping: 0 # 4.15

Threads: 1

SigQ: 0/3067

SigPnd: 0000000000000000

ShdPnd: 0000000000000000

SigBlk: 0000000000010000

SigIgn: 0000000000384004

SigCgt: 000000004b813efb

CapInh: 0000000000000000

CapPrm: 0000000000000000

CapEff: 0000000000000000

CapBnd: ffffffffffffffff

CapAmb: 0000000000000000

NoNewPrivs: 0

Seccomp: 0

Speculation\_Store\_Bypass: vulnerable

Cpus\_allowed: 00000001

Cpus\_allowed\_list: 0

Mems\_allowed: 1

Mems\_allowed\_list: 0

voluntary\_ctxt\_switches: 150

nonvoluntary\_ctxt\_switches: 545

The fields are as follows:

*Name* Command run by this process. Strings longer than

**TASK\_COMM\_LEN** (16) characters (including the

terminating null byte) are silently truncated.

*Umask* Process umask, expressed in octal with a leading

zero; see [umask(2)](https://man7.org/linux/man-pages/man2/umask.2.html). (Since Linux 4.7.)

*State* Current state of the process. One of "R

(running)", "S (sleeping)", "D (disk sleep)", "T

(stopped)", "t (tracing stop)", "Z (zombie)", or "X

(dead)".

*Tgid* Thread group ID (i.e., Process ID).

*Ngid* NUMA group ID (0 if none; since Linux 3.13).

*Pid* Thread ID (see [gettid(2)](https://man7.org/linux/man-pages/man2/gettid.2.html)).

*PPid* PID of parent process.

*TracerPid*

PID of process tracing this process (0 if not being

traced).

*Uid*, *Gid*

Real, effective, saved set, and filesystem UIDs

(GIDs).

*FDSize* Number of file descriptor slots currently

allocated.

*Groups* Supplementary group list.

*NStgid* Thread group ID (i.e., PID) in each of the PID

namespaces of which *[pid]* is a member. The

leftmost entry shows the value with respect to the

PID namespace of the process that mounted this

procfs (or the root namespace if mounted by the

kernel), followed by the value in successively

nested inner namespaces. (Since Linux 4.1.)

*NSpid* Thread ID in each of the PID namespaces of which

*[pid]* is a member. The fields are ordered as for

*NStgid*. (Since Linux 4.1.)

*NSpgid* Process group ID in each of the PID namespaces of

which *[pid]* is a member. The fields are ordered as

for *NStgid*. (Since Linux 4.1.)

*NSsid* descendant namespace session ID hierarchy Session

ID in each of the PID namespaces of which *[pid]* is

a member. The fields are ordered as for *NStgid*.

(Since Linux 4.1.)

*VmPeak* Peak virtual memory size.

*VmSize* Virtual memory size.

*VmLck* Locked memory size (see [mlock(2)](https://man7.org/linux/man-pages/man2/mlock.2.html)).

*VmPin* Pinned memory size (since Linux 3.2). These are

pages that can't be moved because something needs

to directly access physical memory.

*VmHWM* Peak resident set size ("high water mark"). This

value is inaccurate; see */proc/[pid]/statm* above.

*VmRSS* Resident set size. Note that the value here is the

sum of *RssAnon*, *RssFile*, and *RssShmem*. This value

is inaccurate; see */proc/[pid]/statm* above.

*RssAnon*

Size of resident anonymous memory. (since Linux

4.5). This value is inaccurate; see

*/proc/[pid]/statm* above.

*RssFile*

Size of resident file mappings. (since Linux 4.5).

This value is inaccurate; see */proc/[pid]/statm*

above.

*RssShmem*

Size of resident shared memory (includes System V

shared memory, mappings from [tmpfs(5)](https://man7.org/linux/man-pages/man5/tmpfs.5.html), and shared

anonymous mappings). (since Linux 4.5).

*VmData*, *VmStk*, *VmExe*

Size of data, stack, and text segments. This value

is inaccurate; see */proc/[pid]/statm* above.

*VmLib* Shared library code size.

*VmPTE* Page table entries size (since Linux 2.6.10).

*VmPMD* Size of second-level page tables (added in Linux

4.0; removed in Linux 4.15).

*VmSwap* Swapped-out virtual memory size by anonymous

private pages; shmem swap usage is not included

(since Linux 2.6.34). This value is inaccurate;

see */proc/[pid]/statm* above.

*HugetlbPages*

Size of hugetlb memory portions (since Linux 4.4).

*CoreDumping*

Contains the value 1 if the process is currently

dumping core, and 0 if it is not (since Linux

4.15). This information can be used by a

monitoring process to avoid killing a process that

is currently dumping core, which could result in a

corrupted core dump file.

*Threads*

Number of threads in process containing this

thread.

*SigQ* This field contains two slash-separated numbers

that relate to queued signals for the real user ID

of this process. The first of these is the number

of currently queued signals for this real user ID,

and the second is the resource limit on the number

of queued signals for this process (see the

description of **RLIMIT\_SIGPENDING** in [getrlimit(2)](https://man7.org/linux/man-pages/man2/getrlimit.2.html)).

*SigPnd*, *ShdPnd*

Mask (expressed in hexadecimal) of signals pending

for thread and for process as a whole (see

[pthreads(7)](https://man7.org/linux/man-pages/man7/pthreads.7.html) and [signal(7)](https://man7.org/linux/man-pages/man7/signal.7.html)).

*SigBlk*, *SigIgn*, *SigCgt*

Masks (expressed in hexadecimal) indicating signals

being blocked, ignored, and caught (see [signal(7)](https://man7.org/linux/man-pages/man7/signal.7.html)).

*CapInh*, *CapPrm*, *CapEff*

Masks (expressed in hexadecimal) of capabilities

enabled in inheritable, permitted, and effective

sets (see [capabilities(7)](https://man7.org/linux/man-pages/man7/capabilities.7.html)).

*CapBnd* Capability bounding set, expressed in hexadecimal

(since Linux 2.6.26, see [capabilities(7)](https://man7.org/linux/man-pages/man7/capabilities.7.html)).

*CapAmb* Ambient capability set, expressed in hexadecimal

(since Linux 4.3, see [capabilities(7)](https://man7.org/linux/man-pages/man7/capabilities.7.html)).

*NoNewPrivs*

Value of the *no\_new\_privs* bit (since Linux 4.10,

see [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html)).

*Seccomp*

Seccomp mode of the process (since Linux 3.8, see

[seccomp(2)](https://man7.org/linux/man-pages/man2/seccomp.2.html)). 0 means **SECCOMP\_MODE\_DISABLED**; 1

means **SECCOMP\_MODE\_STRICT**; 2 means

**SECCOMP\_MODE\_FILTER**. This field is provided only

if the kernel was built with the **CONFIG\_SECCOMP**

kernel configuration option enabled.

*Speculation\_Store\_Bypass*

Speculation flaw mitigation state (since Linux

4.17, see [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html)).

*Cpus\_allowed*

Hexadecimal mask of CPUs on which this process may

run (since Linux 2.6.24, see [cpuset(7)](https://man7.org/linux/man-pages/man7/cpuset.7.html)).

*Cpus\_allowed\_list*

Same as previous, but in "list format" (since Linux

2.6.26, see [cpuset(7)](https://man7.org/linux/man-pages/man7/cpuset.7.html)).

*Mems\_allowed*

Mask of memory nodes allowed to this process (since

Linux 2.6.24, see [cpuset(7)](https://man7.org/linux/man-pages/man7/cpuset.7.html)).

*Mems\_allowed\_list*

Same as previous, but in "list format" (since Linux

2.6.26, see [cpuset(7)](https://man7.org/linux/man-pages/man7/cpuset.7.html)).

*voluntary\_ctxt\_switches*, *nonvoluntary\_ctxt\_switches*

Number of voluntary and involuntary context

switches (since Linux 2.6.23).

*/proc/[pid]/syscall* (since Linux 2.6.27)

This file exposes the system call number and argument

registers for the system call currently being executed by

the process, followed by the values of the stack pointer

and program counter registers. The values of all six

argument registers are exposed, although most system calls

use fewer registers.

If the process is blocked, but not in a system call, then

the file displays -1 in place of the system call number,

followed by just the values of the stack pointer and

program counter. If process is not blocked, then the file

contains just the string "running".

This file is present only if the kernel was configured

with **CONFIG\_HAVE\_ARCH\_TRACEHOOK**.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_ATTACH\_FSCREDS** check; see

[ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

***/proc/[pid]/task* (since Linux 2.6.0)**

This is a directory that contains one subdirectory for

each thread in the process. The name of each subdirectory

is the numerical thread ID (*[tid]*) of the thread (see

[gettid(2)](https://man7.org/linux/man-pages/man2/gettid.2.html)).

Within each of these subdirectories, there is a set of

files with the same names and contents as under the

*/proc/[pid]* directories. For attributes that are shared

by all threads, the contents for each of the files under

the *task/[tid]* subdirectories will be the same as in the

corresponding file in the parent */proc/[pid]* directory

(e.g., in a multithreaded process, all of the

*task/[tid]/cwd* files will have the same value as the

*/proc/[pid]/cwd* file in the parent directory, since all of

the threads in a process share a working directory). For

attributes that are distinct for each thread, the

corresponding files under *task/[tid]* may have different

values (e.g., various fields in each of the

*task/[tid]/status* files may be different for each thread),

or they might not exist in */proc/[pid]* at all.

In a multithreaded process, the contents of the

*/proc/[pid]/task* directory are not available if the main

thread has already terminated (typically by calling

[pthread\_exit(3)](https://man7.org/linux/man-pages/man3/pthread_exit.3.html)).

*/proc/[pid]/task/[tid]/children* (since Linux 3.5)

A space-separated list of child tasks of this task. Each

child task is represented by its TID.

This option is intended for use by the checkpoint-restore

(CRIU) system, and reliably provides a list of children

only if all of the child processes are stopped or frozen.

It does not work properly if children of the target task

exit while the file is being read! Exiting children may

cause non-exiting children to be omitted from the list.

This makes this interface even more unreliable than

classic PID-based approaches if the inspected task and its

children aren't frozen, and most code should probably not

use this interface.

Until Linux 4.2, the presence of this file was governed by

the **CONFIG\_CHECKPOINT\_RESTORE** kernel configuration option.

Since Linux 4.2, it is governed by the

**CONFIG\_PROC\_CHILDREN** option.

*/proc/[pid]/timers* (since Linux 3.10)

A list of the POSIX timers for this process. Each timer

is listed with a line that starts with the string "ID:".

For example:

ID: 1

signal: 60/00007fff86e452a8

notify: signal/pid.2634

ClockID: 0

ID: 0

signal: 60/00007fff86e452a8

notify: signal/pid.2634

ClockID: 1

The lines shown for each timer have the following

meanings:

*ID* The ID for this timer. This is not the same as the

timer ID returned by [timer\_create(2)](https://man7.org/linux/man-pages/man2/timer_create.2.html); rather, it is

the same kernel-internal ID that is available via

the *si\_timerid* field of the *siginfo\_t* structure

(see [sigaction(2)](https://man7.org/linux/man-pages/man2/sigaction.2.html)).

*signal* This is the signal number that this timer uses to

deliver notifications followed by a slash, and then

the *sigev\_value* value supplied to the signal

handler. Valid only for timers that notify via a

signal.

*notify* The part before the slash specifies the mechanism

that this timer uses to deliver notifications, and

is one of "thread", "signal", or "none".

Immediately following the slash is either the

string "tid" for timers with **SIGEV\_THREAD\_ID**

notification, or "pid" for timers that notify by

other mechanisms. Following the "." is the PID of

the process (or the kernel thread ID of the thread)

that will be delivered a signal if the timer

delivers notifications via a signal.

*ClockID*

This field identifies the clock that the timer uses

for measuring time. For most clocks, this is a

number that matches one of the user-space **CLOCK\_\***

constants exposed via *<time.h>*.

**CLOCK\_PROCESS\_CPUTIME\_ID** timers display with a

value of -6 in this field. **CLOCK\_THREAD\_CPUTIME\_ID**

timers display with a value of -2 in this field.

This file is available only when the kernel was configured

with **CONFIG\_CHECKPOINT\_RESTORE**.

*/proc/[pid]/timerslack\_ns* (since Linux 4.6)

This file exposes the process's "current" timer slack

value, expressed in nanoseconds. The file is writable,

allowing the process's timer slack value to be changed.

Writing 0 to this file resets the "current" timer slack to

the "default" timer slack value. For further details, see

the discussion of **PR\_SET\_TIMERSLACK** in [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html).

Initially, permission to access this file was governed by

a ptrace access mode **PTRACE\_MODE\_ATTACH\_FSCREDS** check (see

[ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html)). However, this was subsequently deemed too

strict a requirement (and had the side effect that

requiring a process to have the **CAP\_SYS\_PTRACE** capability

would also allow it to view and change any process's

memory). Therefore, since Linux 4.9, only the (weaker)

**CAP\_SYS\_NICE** capability is required to access this file.

*/proc/[pid]/uid\_map* (since Linux 3.5)

See [user\_namespaces(7)](https://man7.org/linux/man-pages/man7/user_namespaces.7.html).

***/proc/[pid]/wchan* (since Linux 2.6.0)**

The symbolic name corresponding to the location in the

kernel where the process is sleeping.

Permission to access this file is governed by a ptrace

access mode **PTRACE\_MODE\_READ\_FSCREDS** check; see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

Символическое имя, соответствующее местоположению в ядро, в котором процесс спит.

Разрешение на доступ к этому файлу регулируется ptrace проверка режима доступа PTRACE\_MODE\_READ\_FSCREDS; см. ptrace(2).

***/proc/[tid]***

There is a numerical subdirectory for each running thread

that is not a thread group leader (i.e., a thread whose

thread ID is not the same as its process ID); the

subdirectory is named by the thread ID. Each one of these

subdirectories contains files and subdirectories exposing

information about the thread with the thread ID *tid*. The

contents of these directories are the same as the

corresponding */proc/[pid]/task/[tid]* directories.

The */proc/[tid]* subdirectories are *not* visible when

iterating through */proc* with [getdents(2)](https://man7.org/linux/man-pages/man2/getdents.2.html) (and thus are *not*

visible when one uses [ls(1)](https://man7.org/linux/man-pages/man1/ls.1.html) to view the contents of

*/proc*). However, the pathnames of these directories are

visible to (i.e., usable as arguments in) system calls

that operate on pathnames.

*/proc/apm*

Advanced power management version and battery information

when **CONFIG\_APM** is defined at kernel compilation time.

*/proc/buddyinfo*

This file contains information which is used for

diagnosing memory fragmentation issues. Each line starts

with the identification of the node and the name of the

zone which together identify a memory region. This is

then followed by the count of available chunks of a

certain order in which these zones are split. The size in

bytes of a certain order is given by the formula:

(2^order) \* PAGE\_SIZE

The binary buddy allocator algorithm inside the kernel

will split one chunk into two chunks of a smaller order

(thus with half the size) or combine two contiguous chunks

into one larger chunk of a higher order (thus with double

the size) to satisfy allocation requests and to counter

memory fragmentation. The order matches the column

number, when starting to count at zero.

For example on an x86-64 system:

Node 0, zone DMA 1 1 1 0 2 1 1 0 1 1 3

Node 0, zone DMA32 65 47 4 81 52 28 13 10 5 1 404

Node 0, zone Normal 216 55 189 101 84 38 37 27 5 3 587

In this example, there is one node containing three zones

and there are 11 different chunk sizes. If the page size

is 4 kilobytes, then the first zone called *DMA* (on x86 the

first 16 megabyte of memory) has 1 chunk of 4 kilobytes

(order 0) available and has 3 chunks of 4 megabytes (order

10) available.

If the memory is heavily fragmented, the counters for

higher order chunks will be zero and allocation of large

contiguous areas will fail.

Further information about the zones can be found in

*/proc/zoneinfo*.

*/proc/bus*

Contains subdirectories for installed busses.

*/proc/bus/pccard*

Subdirectory for PCMCIA devices when **CONFIG\_PCMCIA** is set

at kernel compilation time.

*/proc/bus/pccard/drivers*

*/proc/bus/pci*

Contains various bus subdirectories and pseudo-files

containing information about PCI busses, installed

devices, and device drivers. Some of these files are not

ASCII.

*/proc/bus/pci/devices*

Information about PCI devices. They may be accessed

through [lspci(8)](https://man7.org/linux/man-pages/man8/lspci.8.html) and [setpci(8)](https://man7.org/linux/man-pages/man8/setpci.8.html).

*/proc/cgroups* (since Linux 2.6.24)

See [cgroups(7)](https://man7.org/linux/man-pages/man7/cgroups.7.html).

*/proc/cmdline*

Arguments passed to the Linux kernel at boot time. Often

done via a boot manager such as **lilo**(8) or **grub**(8).

*/proc/config.gz* (since Linux 2.6)

This file exposes the configuration options that were used

to build the currently running kernel, in the same format

as they would be shown in the *.config* file that resulted

when configuring the kernel (using *make xconfig*, *make*

*config*, or similar). The file contents are compressed;

view or search them using **zcat**(1) and **zgrep**(1). As long

as no changes have been made to the following file, the

contents of */proc/config.gz* are the same as those provided

by:

cat /lib/modules/$(uname -r)/build/.config

*/proc/config.gz* is provided only if the kernel is

configured with **CONFIG\_IKCONFIG\_PROC**.

*/proc/crypto*

A list of the ciphers provided by the kernel crypto API.

For details, see the kernel *Linux Kernel Crypto API*

documentation available under the kernel source directory

*Documentation/crypto/* (or *Documentation/DocBook* before

4.10; the documentation can be built using a command such

as *make htmldocs* in the root directory of the kernel

source tree).

*/proc/cpuinfo*

This is a collection of CPU and system architecture

dependent items, for each supported architecture a

different list. Two common entries are *processor* which

gives CPU number and *bogomips*; a system constant that is

calculated during kernel initialization. SMP machines

have information for each CPU. The [lscpu(1)](https://man7.org/linux/man-pages/man1/lscpu.1.html) command

gathers its information from this file.

*/proc/devices*

Text listing of major numbers and device groups. This can

be used by MAKEDEV scripts for consistency with the

kernel.

*/proc/diskstats* (since Linux 2.5.69)

This file contains disk I/O statistics for each disk

device. See the Linux kernel source file

*Documentation/iostats.txt* for further information.

*/proc/dma*

This is a list of the registered *ISA* DMA (direct memory

access) channels in use.

*/proc/driver*

Empty subdirectory.

*/proc/execdomains*

List of the execution domains (ABI personalities).

*/proc/fb*

Frame buffer information when **CONFIG\_FB** is defined during

kernel compilation.

***/proc/filesystems***

A text listing of the filesystems which are supported by

the kernel, namely filesystems which were compiled into

the kernel or whose kernel modules are currently loaded.

(See also [filesystems(5)](https://man7.org/linux/man-pages/man5/filesystems.5.html).) If a filesystem is marked with

"nodev", this means that it does not require a block

device to be mounted (e.g., virtual filesystem, network

filesystem).

Incidentally, this file may be used by [mount(8)](https://man7.org/linux/man-pages/man8/mount.8.html) when no

filesystem is specified and it didn't manage to determine

the filesystem type. Then filesystems contained in this

file are tried (excepted those that are marked with

"nodev").

***/proc/fs***

Contains subdirectories that in turn contain files with

information about (certain) mounted filesystems.

*/proc/ide*

This directory exists on systems with the IDE bus. There

are directories for each IDE channel and attached device.

Files include:

cache buffer size in KB

capacity number of sectors

driver driver version

geometry physical and logical geometry

identify in hexadecimal

media media type

model manufacturer's model number

settings drive settings

smart\_thresholds IDE disk management thresholds (in hex)

smart\_values IDE disk management values (in hex)

The [hdparm(8)](https://man7.org/linux/man-pages/man8/hdparm.8.html) utility provides access to this information

in a friendly format.

***/proc/interrupts***

This is used to record the number of interrupts per CPU

per IO device. Since Linux 2.6.24, for the i386 and

x86-64 architectures, at least, this also includes

interrupts internal to the system (that is, not associated

with a device as such), such as NMI (nonmaskable

interrupt), LOC (local timer interrupt), and for SMP

systems, TLB (TLB flush interrupt), RES (rescheduling

interrupt), CAL (remote function call interrupt), and

possibly others. Very easy to read formatting, done in

ASCII.

*/proc/iomem*

I/O memory map in Linux 2.4.

*/proc/ioports*

This is a list of currently registered Input-Output port

regions that are in use.

*/proc/kallsyms* (since Linux 2.5.71)

This holds the kernel exported symbol definitions used by

the **modules**(X) tools to dynamically link and bind loadable

modules. In Linux 2.5.47 and earlier, a similar file with

slightly different syntax was named *ksyms*.

*/proc/kcore*

This file represents the physical memory of the system and

is stored in the ELF core file format. With this pseudo-

file, and an unstripped kernel (*/usr/src/linux/vmlinux*)

binary, GDB can be used to examine the current state of

any kernel data structures.

The total length of the file is the size of physical

memory (RAM) plus 4 KiB.

*/proc/keys* (since Linux 2.6.10)

See [keyrings(7)](https://man7.org/linux/man-pages/man7/keyrings.7.html).

*/proc/key-users* (since Linux 2.6.10)

See [keyrings(7)](https://man7.org/linux/man-pages/man7/keyrings.7.html).

*/proc/kmsg*

This file can be used instead of the [syslog(2)](https://man7.org/linux/man-pages/man2/syslog.2.html) system call

to read kernel messages. A process must have superuser

privileges to read this file, and only one process should

read this file. This file should not be read if a syslog

process is running which uses the [syslog(2)](https://man7.org/linux/man-pages/man2/syslog.2.html) system call

facility to log kernel messages.

Information in this file is retrieved with the [dmesg(1)](https://man7.org/linux/man-pages/man1/dmesg.1.html)

program.

*/proc/kpagecgroup* (since Linux 4.3)

This file contains a 64-bit inode number of the memory

cgroup each page is charged to, indexed by page frame

number (see the discussion of */proc/[pid]/pagemap*).

The */proc/kpagecgroup* file is present only if the

**CONFIG\_MEMCG** kernel configuration option is enabled.

***/proc/kpagecount* (since Linux 2.6.25)**

This file contains a 64-bit count of the number of times

each physical page frame is mapped, indexed by page frame

number (see the discussion of */proc/[pid]/pagemap*).

The */proc/kpagecount* file is present only if the

**CONFIG\_PROC\_PAGE\_MONITOR** kernel configuration option is

enabled.

*/proc/kpageflags* (since Linux 2.6.25)

This file contains 64-bit masks corresponding to each

physical page frame; it is indexed by page frame number

(see the discussion of */proc/[pid]/pagemap*). The bits are

as follows:

0 - KPF\_LOCKED

1 - KPF\_ERROR

2 - KPF\_REFERENCED

3 - KPF\_UPTODATE

4 - KPF\_DIRTY

5 - KPF\_LRU

6 - KPF\_ACTIVE

7 - KPF\_SLAB

8 - KPF\_WRITEBACK

9 - KPF\_RECLAIM

10 - KPF\_BUDDY

11 - KPF\_MMAP (since Linux 2.6.31)

12 - KPF\_ANON (since Linux 2.6.31)

13 - KPF\_SWAPCACHE (since Linux 2.6.31)

14 - KPF\_SWAPBACKED (since Linux 2.6.31)

15 - KPF\_COMPOUND\_HEAD (since Linux 2.6.31)

16 - KPF\_COMPOUND\_TAIL (since Linux 2.6.31)

17 - KPF\_HUGE (since Linux 2.6.31)

18 - KPF\_UNEVICTABLE (since Linux 2.6.31)

19 - KPF\_HWPOISON (since Linux 2.6.31)

20 - KPF\_NOPAGE (since Linux 2.6.31)

21 - KPF\_KSM (since Linux 2.6.32)

22 - KPF\_THP (since Linux 3.4)

23 - KPF\_BALLOON (since Linux 3.18)

24 - KPF\_ZERO\_PAGE (since Linux 4.0)

25 - KPF\_IDLE (since Linux 4.3)

For further details on the meanings of these bits, see the

kernel source file

*Documentation/admin-guide/mm/pagemap.rst*. Before kernel

2.6.29, **KPF\_WRITEBACK**, **KPF\_RECLAIM**, **KPF\_BUDDY**, and

**KPF\_LOCKED** did not report correctly.

The */proc/kpageflags* file is present only if the

**CONFIG\_PROC\_PAGE\_MONITOR** kernel configuration option is

enabled.

*/proc/ksyms* (Linux 1.1.23–2.5.47)

See */proc/kallsyms*.

*/proc/loadavg*

The first three fields in this file are load average

figures giving the number of jobs in the run queue (state

R) or waiting for disk I/O (state D) averaged over 1, 5,

and 15 minutes. They are the same as the load average

numbers given by [uptime(1)](https://man7.org/linux/man-pages/man1/uptime.1.html) and other programs. The fourth

field consists of two numbers separated by a slash (/).

The first of these is the number of currently runnable

kernel scheduling entities (processes, threads). The

value after the slash is the number of kernel scheduling

entities that currently exist on the system. The fifth

field is the PID of the process that was most recently

created on the system.

*/proc/locks*

This file shows current file locks ([flock(2)](https://man7.org/linux/man-pages/man2/flock.2.html) and [fcntl(2)](https://man7.org/linux/man-pages/man2/fcntl.2.html))

and leases ([fcntl(2)](https://man7.org/linux/man-pages/man2/fcntl.2.html)).

An example of the content shown in this file is the

following:

1: POSIX ADVISORY READ 5433 08:01:7864448 128 128

2: FLOCK ADVISORY WRITE 2001 08:01:7864554 0 EOF

3: FLOCK ADVISORY WRITE 1568 00:2f:32388 0 EOF

4: POSIX ADVISORY WRITE 699 00:16:28457 0 EOF

5: POSIX ADVISORY WRITE 764 00:16:21448 0 0

6: POSIX ADVISORY READ 3548 08:01:7867240 1 1

7: POSIX ADVISORY READ 3548 08:01:7865567 1826 2335

8: OFDLCK ADVISORY WRITE -1 08:01:8713209 128 191

The fields shown in each line are as follows:

(1) The ordinal position of the lock in the list.

(2) The lock type. Values that may appear here include:

**FLOCK** This is a BSD file lock created using [flock(2)](https://man7.org/linux/man-pages/man2/flock.2.html).

**OFDLCK** This is an open file description (OFD) lock

created using [fcntl(2)](https://man7.org/linux/man-pages/man2/fcntl.2.html).

**POSIX** This is a POSIX byte-range lock created using

[fcntl(2)](https://man7.org/linux/man-pages/man2/fcntl.2.html).

(3) Among the strings that can appear here are the

following:

**ADVISORY**

This is an advisory lock.

**MANDATORY**

This is a mandatory lock.

(4) The type of lock. Values that can appear here are:

**READ** This is a POSIX or OFD read lock, or a BSD

shared lock.

**WRITE** This is a POSIX or OFD write lock, or a BSD

exclusive lock.

(5) The PID of the process that owns the lock.

Because OFD locks are not owned by a single process

(since multiple processes may have file descriptors

that refer to the same open file description), the

value -1 is displayed in this field for OFD locks.

(Before kernel 4.14, a bug meant that the PID of the

process that initially acquired the lock was displayed

instead of the value -1.)

(6) Three colon-separated subfields that identify the

major and minor device ID of the device containing the

filesystem where the locked file resides, followed by

the inode number of the locked file.

(7) The byte offset of the first byte of the lock. For

BSD locks, this value is always 0.

(8) The byte offset of the last byte of the lock. **EOF** in

this field means that the lock extends to the end of

the file. For BSD locks, the value shown is always

*EOF*.

Since Linux 4.9, the list of locks shown in */proc/locks* is

filtered to show just the locks for the processes in the

PID namespace (see [pid\_namespaces(7)](https://man7.org/linux/man-pages/man7/pid_namespaces.7.html)) for which the */proc*

filesystem was mounted. (In the initial PID namespace,

there is no filtering of the records shown in this file.)

The [lslocks(8)](https://man7.org/linux/man-pages/man8/lslocks.8.html) command provides a bit more information

about each lock.

*/proc/malloc* (only up to and including Linux 2.2)

This file is present only if **CONFIG\_DEBUG\_MALLOC** was

defined during compilation.

*/proc/meminfo*

This file reports statistics about memory usage on the

system. It is used by [free(1)](https://man7.org/linux/man-pages/man1/free.1.html) to report the amount of

free and used memory (both physical and swap) on the

system as well as the shared memory and buffers used by

the kernel. Each line of the file consists of a parameter

name, followed by a colon, the value of the parameter, and

an option unit of measurement (e.g., "kB"). The list

below describes the parameter names and the format

specifier required to read the field value. Except as

noted below, all of the fields have been present since at

least Linux 2.6.0. Some fields are displayed only if the

kernel was configured with various options; those

dependencies are noted in the list.

*MemTotal* %lu

Total usable RAM (i.e., physical RAM minus a few

reserved bits and the kernel binary code).

*MemFree* %lu

The sum of *LowFree*+*HighFree*.

*MemAvailable* %lu (since Linux 3.14)

An estimate of how much memory is available for

starting new applications, without swapping.

*Buffers* %lu

Relatively temporary storage for raw disk blocks

that shouldn't get tremendously large (20 MB or

so).

*Cached* %lu

In-memory cache for files read from the disk (the

page cache). Doesn't include *SwapCached*.

*SwapCached* %lu

Memory that once was swapped out, is swapped back

in but still also is in the swap file. (If memory

pressure is high, these pages don't need to be

swapped out again because they are already in the

swap file. This saves I/O.)

*Active* %lu

Memory that has been used more recently and usually

not reclaimed unless absolutely necessary.

*Inactive* %lu

Memory which has been less recently used. It is

more eligible to be reclaimed for other purposes.

*Active(anon)* %lu (since Linux 2.6.28)

[To be documented.]

*Inactive(anon)* %lu (since Linux 2.6.28)

[To be documented.]

*Active(file)* %lu (since Linux 2.6.28)

[To be documented.]

*Inactive(file)* %lu (since Linux 2.6.28)

[To be documented.]

*Unevictable* %lu (since Linux 2.6.28)

(From Linux 2.6.28 to 2.6.30,

**CONFIG\_UNEVICTABLE\_LRU** was required.) [To be

documented.]

*Mlocked* %lu (since Linux 2.6.28)

(From Linux 2.6.28 to 2.6.30,

**CONFIG\_UNEVICTABLE\_LRU** was required.) [To be

documented.]

*HighTotal* %lu

(Starting with Linux 2.6.19, **CONFIG\_HIGHMEM** is

required.) Total amount of highmem. Highmem is

all memory above ~860 MB of physical memory.

Highmem areas are for use by user-space programs,

or for the page cache. The kernel must use tricks

to access this memory, making it slower to access

than lowmem.

*HighFree* %lu

(Starting with Linux 2.6.19, **CONFIG\_HIGHMEM** is

required.) Amount of free highmem.

*LowTotal* %lu

(Starting with Linux 2.6.19, **CONFIG\_HIGHMEM** is

required.) Total amount of lowmem. Lowmem is

memory which can be used for everything that

highmem can be used for, but it is also available

for the kernel's use for its own data structures.

Among many other things, it is where everything

from *Slab* is allocated. Bad things happen when

you're out of lowmem.

*LowFree* %lu

(Starting with Linux 2.6.19, **CONFIG\_HIGHMEM** is

required.) Amount of free lowmem.

*MmapCopy* %lu (since Linux 2.6.29)

(**CONFIG\_MMU** is required.) [To be documented.]

*SwapTotal* %lu

Total amount of swap space available.

*SwapFree* %lu

Amount of swap space that is currently unused.

*Dirty* %lu

Memory which is waiting to get written back to the

disk.

*Writeback* %lu

Memory which is actively being written back to the

disk.

*AnonPages* %lu (since Linux 2.6.18)

Non-file backed pages mapped into user-space page

tables.

*Mapped* %lu

Files which have been mapped into memory (with

[mmap(2)](https://man7.org/linux/man-pages/man2/mmap.2.html)), such as libraries.

*Shmem* %lu (since Linux 2.6.32)

Amount of memory consumed in [tmpfs(5)](https://man7.org/linux/man-pages/man5/tmpfs.5.html) filesystems.

*KReclaimable* %lu (since Linux 4.20)

Kernel allocations that the kernel will attempt to

reclaim under memory pressure. Includes

*SReclaimable* (below), and other direct allocations

with a shrinker.

*Slab* %lu

In-kernel data structures cache. (See

[slabinfo(5)](https://man7.org/linux/man-pages/man5/slabinfo.5.html).)

*SReclaimable* %lu (since Linux 2.6.19)

Part of *Slab*, that might be reclaimed, such as

caches.

*SUnreclaim* %lu (since Linux 2.6.19)

Part of *Slab*, that cannot be reclaimed on memory

pressure.

*KernelStack* %lu (since Linux 2.6.32)

Amount of memory allocated to kernel stacks.

*PageTables* %lu (since Linux 2.6.18)

Amount of memory dedicated to the lowest level of

page tables.

*Quicklists* %lu (since Linux 2.6.27)

(**CONFIG\_QUICKLIST** is required.) [To be

documented.]

*NFS\_Unstable* %lu (since Linux 2.6.18)

NFS pages sent to the server, but not yet committed

to stable storage.

*Bounce* %lu (since Linux 2.6.18)

Memory used for block device "bounce buffers".

*WritebackTmp* %lu (since Linux 2.6.26)

Memory used by FUSE for temporary writeback

buffers.

*CommitLimit* %lu (since Linux 2.6.10)

This is the total amount of memory currently

available to be allocated on the system, expressed

in kilobytes. This limit is adhered to only if

strict overcommit accounting is enabled (mode 2 in

*/proc/sys/vm/overcommit\_memory*). The limit is

calculated according to the formula described under

*/proc/sys/vm/overcommit\_memory*. For further

details, see the kernel source file

*Documentation/vm/overcommit-accounting.rst*.

*Committed\_AS* %lu

The amount of memory presently allocated on the

system. The committed memory is a sum of all of

the memory which has been allocated by processes,

even if it has not been "used" by them as of yet.

A process which allocates 1 GB of memory (using

[malloc(3)](https://man7.org/linux/man-pages/man3/malloc.3.html) or similar), but touches only 300 MB of

that memory will show up as using only 300 MB of

memory even if it has the address space allocated

for the entire 1 GB.

This 1 GB is memory which has been "committed" to

by the VM and can be used at any time by the

allocating application. With strict overcommit

enabled on the system (mode 2 in

*/proc/sys/vm/overcommit\_memory*), allocations which

would exceed the *CommitLimit* will not be permitted.

This is useful if one needs to guarantee that

processes will not fail due to lack of memory once

that memory has been successfully allocated.

*VmallocTotal* %lu

Total size of vmalloc memory area.

*VmallocUsed* %lu

Amount of vmalloc area which is used. Since Linux

4.4, this field is no longer calculated, and is

hard coded as 0. See */proc/vmallocinfo*.

*VmallocChunk* %lu

Largest contiguous block of vmalloc area which is

free. Since Linux 4.4, this field is no longer

calculated and is hard coded as 0. See

*/proc/vmallocinfo*.

*HardwareCorrupted* %lu (since Linux 2.6.32)

(**CONFIG\_MEMORY\_FAILURE** is required.) [To be

documented.]

*LazyFree* %lu (since Linux 4.12)

Shows the amount of memory marked by [madvise(2)](https://man7.org/linux/man-pages/man2/madvise.2.html)

**MADV\_FREE**.

*AnonHugePages* %lu (since Linux 2.6.38)

(**CONFIG\_TRANSPARENT\_HUGEPAGE** is required.) Non-

file backed huge pages mapped into user-space page

tables.

*ShmemHugePages* %lu (since Linux 4.8)

(**CONFIG\_TRANSPARENT\_HUGEPAGE** is required.) Memory

used by shared memory (shmem) and [tmpfs(5)](https://man7.org/linux/man-pages/man5/tmpfs.5.html)

allocated with huge pages.

*ShmemPmdMapped* %lu (since Linux 4.8)

(**CONFIG\_TRANSPARENT\_HUGEPAGE** is required.) Shared

memory mapped into user space with huge pages.

*CmaTotal* %lu (since Linux 3.1)

Total CMA (Contiguous Memory Allocator) pages.

(**CONFIG\_CMA** is required.)

*CmaFree* %lu (since Linux 3.1)

Free CMA (Contiguous Memory Allocator) pages.

(**CONFIG\_CMA** is required.)

*HugePages\_Total* %lu

(**CONFIG\_HUGETLB\_PAGE** is required.) The size of the

pool of huge pages.

*HugePages\_Free* %lu

(**CONFIG\_HUGETLB\_PAGE** is required.) The number of

huge pages in the pool that are not yet allocated.

*HugePages\_Rsvd* %lu (since Linux 2.6.17)

(**CONFIG\_HUGETLB\_PAGE** is required.) This is the

number of huge pages for which a commitment to

allocate from the pool has been made, but no

allocation has yet been made. These reserved huge

pages guarantee that an application will be able to

allocate a huge page from the pool of huge pages at

fault time.

*HugePages\_Surp* %lu (since Linux 2.6.24)

(**CONFIG\_HUGETLB\_PAGE** is required.) This is the

number of huge pages in the pool above the value in

*/proc/sys/vm/nr\_hugepages*. The maximum number of

surplus huge pages is controlled by

*/proc/sys/vm/nr\_overcommit\_hugepages*.

*Hugepagesize* %lu

(**CONFIG\_HUGETLB\_PAGE** is required.) The size of

huge pages.

*DirectMap4k* %lu (since Linux 2.6.27)

Number of bytes of RAM linearly mapped by kernel in

4 kB pages. (x86.)

*DirectMap4M* %lu (since Linux 2.6.27)

Number of bytes of RAM linearly mapped by kernel in

4 MB pages. (x86 with **CONFIG\_X86\_64** or

**CONFIG\_X86\_PAE** enabled.)

*DirectMap2M* %lu (since Linux 2.6.27)

Number of bytes of RAM linearly mapped by kernel in

2 MB pages. (x86 with neither **CONFIG\_X86\_64** nor

**CONFIG\_X86\_PAE** enabled.)

*DirectMap1G* %lu (since Linux 2.6.27)

(x86 with **CONFIG\_X86\_64** and

**CONFIG\_X86\_DIRECT\_GBPAGES** enabled.)

***/proc/modules***

A text list of the modules that have been loaded by the

system. See also [lsmod(8)](https://man7.org/linux/man-pages/man8/lsmod.8.html).

***/proc/mounts***

Before kernel 2.4.19, this file was a list of all the

filesystems currently mounted on the system. With the

introduction of per-process mount namespaces in Linux

2.4.19 (see [mount\_namespaces(7)](https://man7.org/linux/man-pages/man7/mount_namespaces.7.html)), this file became a link

to */proc/self/mounts*, which lists the mounts of the

process's own mount namespace. The format of this file is

documented in [fstab(5)](https://man7.org/linux/man-pages/man5/fstab.5.html).

*/proc/mtrr*

Memory Type Range Registers. See the Linux kernel source

file *Documentation/x86/mtrr.txt* (or *Documentation/mtrr.txt*

before Linux 2.6.28) for details.

***/proc/net***

This directory contains various files and subdirectories

containing information about the networking layer. The

files contain ASCII structures and are, therefore,

readable with [cat(1)](https://man7.org/linux/man-pages/man1/cat.1.html). However, the standard [netstat(8)](https://man7.org/linux/man-pages/man8/netstat.8.html)

suite provides much cleaner access to these files.

With the advent of network namespaces, various information

relating to the network stack is virtualized (see

[network\_namespaces(7)](https://man7.org/linux/man-pages/man7/network_namespaces.7.html)). Thus, since Linux 2.6.25,

*/proc/net* is a symbolic link to the directory

*/proc/self/net*, which contains the same files and

directories as listed below. However, these files and

directories now expose information for the network

namespace of which the process is a member.

*/proc/net/arp*

This holds an ASCII readable dump of the kernel ARP table

used for address resolutions. It will show both

dynamically learned and preprogrammed ARP entries. The

format is:

IP address HW type Flags HW address Mask Device

192.168.0.50 0x1 0x2 00:50:BF:25:68:F3 \* eth0

192.168.0.250 0x1 0xc 00:00:00:00:00:00 \* eth0

Here "IP address" is the IPv4 address of the machine and

the "HW type" is the hardware type of the address from

RFC 826. The flags are the internal flags of the ARP

structure (as defined in */usr/include/linux/if\_arp.h*) and

the "HW address" is the data link layer mapping for that

IP address if it is known.

***/proc/net/dev***

The dev pseudo-file contains network device status

information. This gives the number of received and sent

packets, the number of errors and collisions and other

basic statistics. These are used by the [ifconfig(8)](https://man7.org/linux/man-pages/man8/ifconfig.8.html)

program to report device status. The format is:

Inter-| Receive | Transmit

face |bytes packets errs drop fifo frame compressed multicast|bytes packets errs drop fifo colls carrier compressed

lo: 2776770 11307 0 0 0 0 0 0 2776770 11307 0 0 0 0 0 0

eth0: 1215645 2751 0 0 0 0 0 0 1782404 4324 0 0 0 427 0 0

ppp0: 1622270 5552 1 0 0 0 0 0 354130 5669 0 0 0 0 0 0

tap0: 7714 81 0 0 0 0 0 0 7714 81 0 0 0 0 0 0

*/proc/net/dev\_mcast*

Defined in */usr/src/linux/net/core/dev\_mcast.c*:

indx interface\_name dmi\_u dmi\_g dmi\_address

2 eth0 1 0 01005e000001

3 eth1 1 0 01005e000001

4 eth2 1 0 01005e000001

*/proc/net/igmp*

Internet Group Management Protocol. Defined in

*/usr/src/linux/net/core/igmp.c*.

*/proc/net/rarp*

This file uses the same format as the *arp* file and

contains the current reverse mapping database used to

provide [rarp(8)](https://man7.org/linux/man-pages/man8/rarp.8.html) reverse address lookup services. If RARP

is not configured into the kernel, this file will not be

present.

*/proc/net/raw*

Holds a dump of the RAW socket table. Much of the

information is not of use apart from debugging. The "sl"

value is the kernel hash slot for the socket, the

"local\_address" is the local address and protocol number

pair. "St" is the internal status of the socket. The

"tx\_queue" and "rx\_queue" are the outgoing and incoming

data queue in terms of kernel memory usage. The "tr",

"tm->when", and "rexmits" fields are not used by RAW. The

"uid" field holds the effective UID of the creator of the

socket.

*/proc/net/snmp*

This file holds the ASCII data needed for the IP, ICMP,

TCP, and UDP management information bases for an SNMP

agent.

*/proc/net/tcp*

Holds a dump of the TCP socket table. Much of the

information is not of use apart from debugging. The "sl"

value is the kernel hash slot for the socket, the

"local\_address" is the local address and port number pair.

The "rem\_address" is the remote address and port number

pair (if connected). "St" is the internal status of the

socket. The "tx\_queue" and "rx\_queue" are the outgoing

and incoming data queue in terms of kernel memory usage.

The "tr", "tm->when", and "rexmits" fields hold internal

information of the kernel socket state and are useful only

for debugging. The "uid" field holds the effective UID of

the creator of the socket.

*/proc/net/udp*

Holds a dump of the UDP socket table. Much of the

information is not of use apart from debugging. The "sl"

value is the kernel hash slot for the socket, the

"local\_address" is the local address and port number pair.

The "rem\_address" is the remote address and port number

pair (if connected). "St" is the internal status of the

socket. The "tx\_queue" and "rx\_queue" are the outgoing

and incoming data queue in terms of kernel memory usage.

The "tr", "tm->when", and "rexmits" fields are not used by

UDP. The "uid" field holds the effective UID of the

creator of the socket. The format is:

sl local\_address rem\_address st tx\_queue rx\_queue tr rexmits tm->when uid

1: 01642C89:0201 0C642C89:03FF 01 00000000:00000001 01:000071BA 00000000 0

1: 00000000:0801 00000000:0000 0A 00000000:00000000 00:00000000 6F000100 0

1: 00000000:0201 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0

*/proc/net/unix*

Lists the UNIX domain sockets present within the system

and their status. The format is:

Num RefCount Protocol Flags Type St Inode Path

0: 00000002 00000000 00000000 0001 03 42

1: 00000001 00000000 00010000 0001 01 1948 /dev/printer

The fields are as follows:

*Num*: the kernel table slot number.

*RefCount*:

the number of users of the socket.

*Protocol*:

currently always 0.

*Flags*: the internal kernel flags holding the status of the

socket.

*Type*: the socket type. For **SOCK\_STREAM** sockets, this is

0001; for **SOCK\_DGRAM** sockets, it is 0002; and for

**SOCK\_SEQPACKET** sockets, it is 0005.

*St*: the internal state of the socket.

*Inode*: the inode number of the socket.

*Path*: the bound pathname (if any) of the socket. Sockets

in the abstract namespace are included in the list,

and are shown with a *Path* that commences with the

character '@'.

*/proc/net/netfilter/nfnetlink\_queue*

This file contains information about netfilter user-space

queueing, if used. Each line represents a queue. Queues

that have not been subscribed to by user space are not

shown.

1 4207 0 2 65535 0 0 0 1

(1) (2) (3)(4) (5) (6) (7) (8)

The fields in each line are:

(1) The ID of the queue. This matches what is specified

in the **--queue-num** or **--queue-balance** options to the

[iptables(8)](https://man7.org/linux/man-pages/man8/iptables.8.html) NFQUEUE target. See

[iptables-extensions(8)](https://man7.org/linux/man-pages/man8/iptables-extensions.8.html) for more information.

(2) The netlink port ID subscribed to the queue.

(3) The number of packets currently queued and waiting to

be processed by the application.

(4) The copy mode of the queue. It is either 1 (metadata

only) or 2 (also copy payload data to user space).

(5) Copy range; that is, how many bytes of packet payload

should be copied to user space at most.

(6) queue dropped. Number of packets that had to be

dropped by the kernel because too many packets are

already waiting for user space to send back the

mandatory accept/drop verdicts.

(7) queue user dropped. Number of packets that were

dropped within the netlink subsystem. Such drops

usually happen when the corresponding socket buffer

is full; that is, user space is not able to read

messages fast enough.

(8) sequence number. Every queued packet is associated

with a (32-bit) monotonically increasing sequence

number. This shows the ID of the most recent packet

queued.

The last number exists only for compatibility reasons and

is always 1.

***/proc/partitions***

Contains the major and minor numbers of each partition as

well as the number of 1024-byte blocks and the partition

name.

*/proc/pci*

This is a listing of all PCI devices found during kernel

initialization and their configuration.

This file has been deprecated in favor of a new */proc*

interface for PCI (*/proc/bus/pci*). It became optional in

Linux 2.2 (available with **CONFIG\_PCI\_OLD\_PROC** set at

kernel compilation). It became once more nonoptionally

enabled in Linux 2.4. Next, it was deprecated in Linux

2.6 (still available with **CONFIG\_PCI\_LEGACY\_PROC** set), and

finally removed altogether since Linux 2.6.17.

*/proc/profile* (since Linux 2.4)

This file is present only if the kernel was booted with

the *profile=1* command-line option. It exposes kernel

profiling information in a binary format for use by

**readprofile**(1). Writing (e.g., an empty string) to this

file resets the profiling counters; on some architectures,

writing a binary integer "profiling multiplier" of size

*sizeof(int)* sets the profiling interrupt frequency.

*/proc/scsi*

A directory with the *scsi* mid-level pseudo-file and

various SCSI low-level driver directories, which contain a

file for each SCSI host in this system, all of which give

the status of some part of the SCSI IO subsystem. These

files contain ASCII structures and are, therefore,

readable with [cat(1)](https://man7.org/linux/man-pages/man1/cat.1.html).

You can also write to some of the files to reconfigure the

subsystem or switch certain features on or off.

*/proc/scsi/scsi*

This is a listing of all SCSI devices known to the kernel.

The listing is similar to the one seen during bootup.

scsi currently supports only the *add-single-device* command

which allows root to add a hotplugged device to the list

of known devices.

The command

echo 'scsi add-single-device 1 0 5 0' > /proc/scsi/scsi

will cause host scsi1 to scan on SCSI channel 0 for a

device on ID 5 LUN 0. If there is already a device known

on this address or the address is invalid, an error will

be returned.

*/proc/scsi/[drivername]*

*[drivername]* can currently be NCR53c7xx, aha152x, aha1542,

aha1740, aic7xxx, buslogic, eata\_dma, eata\_pio, fdomain,

in2000, pas16, qlogic, scsi\_debug, seagate, t128, u15-24f,

ultrastore, or wd7000. These directories show up for all

drivers that registered at least one SCSI HBA. Every

directory contains one file per registered host. Every

host-file is named after the number the host was assigned

during initialization.

Reading these files will usually show driver and host

configuration, statistics, and so on.

Writing to these files allows different things on

different hosts. For example, with the *latency* and

*nolatency* commands, root can switch on and off command

latency measurement code in the eata\_dma driver. With the

*lockup* and *unlock* commands, root can control bus lockups

simulated by the scsi\_debug driver.

*/proc/self*

This directory refers to the process accessing the */proc*

filesystem, and is identical to the */proc* directory named

by the process ID of the same process.

*/proc/slabinfo*

Information about kernel caches. See [slabinfo(5)](https://man7.org/linux/man-pages/man5/slabinfo.5.html) for

details.

***/proc/stat***

kernel/system statistics. Varies with architecture.

Common entries include:

*cpu 10132153 290696 3084719 46828483 16683 0 25195 0*

*175628 0*

*cpu0 1393280 32966 572056 13343292 6130 0 17875 0 23933 0*

The amount of time, measured in units of USER\_HZ

(1/100ths of a second on most architectures, use

*sysconf(\_SC\_CLK\_TCK)* to obtain the right value),

that the system ("cpu" line) or the specific CPU

("cpu*N*" line) spent in various states:

*user* (1) Time spent in user mode.

*nice* (2) Time spent in user mode with low

priority (nice).

*system* (3) Time spent in system mode.

*idle* (4) Time spent in the idle task. This value

should be USER\_HZ times the second entry in

the */proc/uptime* pseudo-file.

*iowait* (since Linux 2.5.41)

(5) Time waiting for I/O to complete. This

value is not reliable, for the following

reasons:

1. The CPU will not wait for I/O to

complete; iowait is the time that a task

is waiting for I/O to complete. When a

CPU goes into idle state for outstanding

task I/O, another task will be scheduled

on this CPU.

2. On a multi-core CPU, the task waiting for

I/O to complete is not running on any

CPU, so the iowait of each CPU is

difficult to calculate.

3. The value in this field may *decrease* in

certain conditions.

*irq* (since Linux 2.6.0)

(6) Time servicing interrupts.

*softirq* (since Linux 2.6.0)

(7) Time servicing softirqs.

*steal* (since Linux 2.6.11)

(8) Stolen time, which is the time spent in

other operating systems when running in a

virtualized environment

*guest* (since Linux 2.6.24)

(9) Time spent running a virtual CPU for

guest operating systems under the control of

the Linux kernel.

*guest\_nice* (since Linux 2.6.33)

(10) Time spent running a niced guest

(virtual CPU for guest operating systems

under the control of the Linux kernel).

*page 5741 1808*

The number of pages the system paged in and the

number that were paged out (from disk).

*swap 1 0*

The number of swap pages that have been brought in

and out.

*intr 1462898*

This line shows counts of interrupts serviced since

boot time, for each of the possible system

interrupts. The first column is the total of all

interrupts serviced including unnumbered

architecture specific interrupts; each subsequent

column is the total for that particular numbered

interrupt. Unnumbered interrupts are not shown,

only summed into the total.

*disk\_io: (2,0):(31,30,5764,1,2) (3,0):*...

(major,disk\_idx):(noinfo, read\_io\_ops, blks\_read,

write\_io\_ops, blks\_written)

(Linux 2.4 only)

*ctxt 115315*

The number of context switches that the system

underwent.

*btime 769041601*

boot time, in seconds since the Epoch, 1970-01-01

00:00:00 +0000 (UTC).

*processes 86031*

Number of forks since boot.

*procs\_running 6*

Number of processes in runnable state. (Linux

2.5.45 onward.)

*procs\_blocked 2*

Number of processes blocked waiting for I/O to

complete. (Linux 2.5.45 onward.)

*softirq 229245889 94 60001584 13619 5175704 2471304 28*

*51212741 59130143 0 51240672*

This line shows the number of softirq for all CPUs.

The first column is the total of all softirqs and

each subsequent column is the total for particular

softirq. (Linux 2.6.31 onward.)

*/proc/swaps*

Swap areas in use. See also [swapon(8)](https://man7.org/linux/man-pages/man8/swapon.8.html).

*/proc/sys*

This directory (present since 1.3.57) contains a number of

files and subdirectories corresponding to kernel

variables. These variables can be read and in some cases

modified using the */proc* filesystem, and the (deprecated)

[sysctl(2)](https://man7.org/linux/man-pages/man2/sysctl.2.html) system call.

String values may be terminated by either '\0' or '\n'.

Integer and long values may be written either in decimal

or in hexadecimal notation (e.g., 0x3FFF). When writing

multiple integer or long values, these may be separated by

any of the following whitespace characters: ' ', '\t', or

'\n'. Using other separators leads to the error **EINVAL**.

*/proc/sys/abi* (since Linux 2.4.10)

This directory may contain files with application binary

information. See the Linux kernel source file

*Documentation/sysctl/abi.txt* for more information.

*/proc/sys/debug*

This directory may be empty.

*/proc/sys/dev*

This directory contains device-specific information (e.g.,

*dev/cdrom/info*). On some systems, it may be empty.

*/proc/sys/fs*

This directory contains the files and subdirectories for

kernel variables related to filesystems.

*/proc/sys/fs/aio-max-nr* and */proc/sys/fs/aio-nr* (since Linux

2.6.4)

*aio-nr* is the running total of the number of events

specified by [io\_setup(2)](https://man7.org/linux/man-pages/man2/io_setup.2.html) calls for all currently active

AIO contexts. If *aio-nr* reaches *aio-max-nr*, then

[io\_setup(2)](https://man7.org/linux/man-pages/man2/io_setup.2.html) will fail with the error **EAGAIN**. Raising

*aio-max-nr* does not result in the preallocation or

resizing of any kernel data structures.

*/proc/sys/fs/binfmt\_misc*

Documentation for files in this directory can be found in

the Linux kernel source in the file

*Documentation/admin-guide/binfmt-misc.rst* (or in

*Documentation/binfmt\_misc.txt* on older kernels).

*/proc/sys/fs/dentry-state* (since Linux 2.2)

This file contains information about the status of the

directory cache (dcache). The file contains six numbers,

*nr\_dentry*, *nr\_unused*, *age\_limit* (age in seconds),

*want\_pages* (pages requested by system) and two dummy

values.

\* *nr\_dentry* is the number of allocated dentries (dcache

entries). This field is unused in Linux 2.2.

\* *nr\_unused* is the number of unused dentries.

\* *age\_limit* is the age in seconds after which dcache

entries can be reclaimed when memory is short.

\* *want\_pages* is nonzero when the kernel has called

shrink\_dcache\_pages() and the dcache isn't pruned yet.

*/proc/sys/fs/dir-notify-enable*

This file can be used to disable or enable the *dnotify*

interface described in [fcntl(2)](https://man7.org/linux/man-pages/man2/fcntl.2.html) on a system-wide basis. A

value of 0 in this file disables the interface, and a

value of 1 enables it.

*/proc/sys/fs/dquot-max*

This file shows the maximum number of cached disk quota

entries. On some (2.4) systems, it is not present. If

the number of free cached disk quota entries is very low

and you have some awesome number of simultaneous system

users, you might want to raise the limit.

*/proc/sys/fs/dquot-nr*

This file shows the number of allocated disk quota entries

and the number of free disk quota entries.

*/proc/sys/fs/epoll* (since Linux 2.6.28)

This directory contains the file *max\_user\_watches*, which

can be used to limit the amount of kernel memory consumed

by the *epoll* interface. For further details, see

[epoll(7)](https://man7.org/linux/man-pages/man7/epoll.7.html).

*/proc/sys/fs/file-max*

This file defines a system-wide limit on the number of

open files for all processes. System calls that fail when

encountering this limit fail with the error **ENFILE**. (See

also [setrlimit(2)](https://man7.org/linux/man-pages/man2/setrlimit.2.html), which can be used by a process to set

the per-process limit, **RLIMIT\_NOFILE**, on the number of

files it may open.) If you get lots of error messages in

the kernel log about running out of file handles (open

file descriptions) (look for "VFS: file-max limit <number>

reached"), try increasing this value:

echo 100000 > /proc/sys/fs/file-max

Privileged processes (**CAP\_SYS\_ADMIN**) can override the

*file-max* limit.

*/proc/sys/fs/file-nr*

This (read-only) file contains three numbers: the number

of allocated file handles (i.e., the number of open file

descriptions; see [open(2)](https://man7.org/linux/man-pages/man2/open.2.html)); the number of free file

handles; and the maximum number of file handles (i.e., the

same value as */proc/sys/fs/file-max*). If the number of

allocated file handles is close to the maximum, you should

consider increasing the maximum. Before Linux 2.6, the

kernel allocated file handles dynamically, but it didn't

free them again. Instead the free file handles were kept

in a list for reallocation; the "free file handles" value

indicates the size of that list. A large number of free

file handles indicates that there was a past peak in the

usage of open file handles. Since Linux 2.6, the kernel

does deallocate freed file handles, and the "free file

handles" value is always zero.

*/proc/sys/fs/inode-max* (only present until Linux 2.2)

This file contains the maximum number of in-memory inodes.

This value should be 3–4 times larger than the value in

*file-max*, since *stdin*, *stdout* and network sockets also

need an inode to handle them. When you regularly run out

of inodes, you need to increase this value.

Starting with Linux 2.4, there is no longer a static limit

on the number of inodes, and this file is removed.

*/proc/sys/fs/inode-nr*

This file contains the first two values from *inode-state*.

*/proc/sys/fs/inode-state*

This file contains seven numbers: *nr\_inodes*,

*nr\_free\_inodes*, *preshrink*, and four dummy values (always

zero).

*nr\_inodes* is the number of inodes the system has

allocated. *nr\_free\_inodes* represents the number of free

inodes.

*preshrink* is nonzero when the *nr\_inodes* > *inode-max* and

the system needs to prune the inode list instead of

allocating more; since Linux 2.4, this field is a dummy

value (always zero).

*/proc/sys/fs/inotify* (since Linux 2.6.13)

This directory contains files *max\_queued\_events*,

*max\_user\_instances*, and *max\_user\_watches*, that can be used

to limit the amount of kernel memory consumed by the

*inotify* interface. For further details, see [inotify(7)](https://man7.org/linux/man-pages/man7/inotify.7.html).

*/proc/sys/fs/lease-break-time*

This file specifies the grace period that the kernel

grants to a process holding a file lease ([fcntl(2)](https://man7.org/linux/man-pages/man2/fcntl.2.html)) after

it has sent a signal to that process notifying it that

another process is waiting to open the file. If the lease

holder does not remove or downgrade the lease within this

grace period, the kernel forcibly breaks the lease.

*/proc/sys/fs/leases-enable*

This file can be used to enable or disable file leases

([fcntl(2)](https://man7.org/linux/man-pages/man2/fcntl.2.html)) on a system-wide basis. If this file contains

the value 0, leases are disabled. A nonzero value enables

leases.

*/proc/sys/fs/mount-max* (since Linux 4.9)

The value in this file specifies the maximum number of

mounts that may exist in a mount namespace. The default

value in this file is 100,000.

*/proc/sys/fs/mqueue* (since Linux 2.6.6)

This directory contains files *msg\_max*, *msgsize\_max*, and

*queues\_max*, controlling the resources used by POSIX

message queues. See [mq\_overview(7)](https://man7.org/linux/man-pages/man7/mq_overview.7.html) for details.

*/proc/sys/fs/nr\_open* (since Linux 2.6.25)

This file imposes a ceiling on the value to which the

**RLIMIT\_NOFILE** resource limit can be raised (see

[getrlimit(2)](https://man7.org/linux/man-pages/man2/getrlimit.2.html)). This ceiling is enforced for both

unprivileged and privileged process. The default value in

this file is 1048576. (Before Linux 2.6.25, the ceiling

for **RLIMIT\_NOFILE** was hard-coded to the same value.)

*/proc/sys/fs/overflowgid* and */proc/sys/fs/overflowuid*

These files allow you to change the value of the fixed UID

and GID. The default is 65534. Some filesystems support

only 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 the overflow value before being

written to disk.

*/proc/sys/fs/pipe-max-size* (since Linux 2.6.35)

See [pipe(7)](https://man7.org/linux/man-pages/man7/pipe.7.html).

*/proc/sys/fs/pipe-user-pages-hard* (since Linux 4.5)

See [pipe(7)](https://man7.org/linux/man-pages/man7/pipe.7.html).

*/proc/sys/fs/pipe-user-pages-soft* (since Linux 4.5)

See [pipe(7)](https://man7.org/linux/man-pages/man7/pipe.7.html).

*/proc/sys/fs/protected\_fifos* (since Linux 4.19)

The value in this file is/can be set to one of the

following:

0 Writing to FIFOs is unrestricted.

1 Don't allow **O\_CREAT open**(2) on FIFOs that the caller

doesn't own in world-writable sticky directories,

unless the FIFO is owned by the owner of the

directory.

2 As for the value 1, but the restriction also applies

to group-writable sticky directories.

The intent of the above protections is to avoid

unintentional writes to an attacker-controlled FIFO when a

program expected to create a regular file.

*/proc/sys/fs/protected\_hardlinks* (since Linux 3.6)

When the value in this file is 0, no restrictions are

placed on the creation of hard links (i.e., this is the

historical behavior before Linux 3.6). When the value in

this file is 1, a hard link can be created to a target

file only if one of the following conditions is true:

\* The calling process has the **CAP\_FOWNER** capability in

its user namespace and the file UID has a mapping in

the namespace.

\* The filesystem UID of the process creating the link

matches the owner (UID) of the target file (as

described in [credentials(7)](https://man7.org/linux/man-pages/man7/credentials.7.html), a process's filesystem UID

is normally the same as its effective UID).

\* All of the following conditions are true:

• the target is a regular file;

• the target file does not have its set-user-ID mode

bit enabled;

• the target file does not have both its set-group-ID

and group-executable mode bits enabled; and

• the caller has permission to read and write the

target file (either via the file's permissions mask

or because it has suitable capabilities).

The default value in this file is 0. Setting the value to

1 prevents a longstanding class of security issues caused

by hard-link-based time-of-check, time-of-use races, most

commonly seen in world-writable directories such as */tmp*.

The common method of exploiting this flaw is to cross

privilege boundaries when following a given hard link

(i.e., a root process follows a hard link created by

another user). Additionally, on systems without separated

partitions, this stops unauthorized users from "pinning"

vulnerable set-user-ID and set-group-ID files against

being upgraded by the administrator, or linking to special

files.

*/proc/sys/fs/protected\_regular* (since Linux 4.19)

The value in this file is/can be set to one of the

following:

0 Writing to regular files is unrestricted.

1 Don't allow **O\_CREAT open**(2) on regular files that the

caller doesn't own in world-writable sticky

directories, unless the regular file is owned by the

owner of the directory.

2 As for the value 1, but the restriction also applies

to group-writable sticky directories.

The intent of the above protections is similar to

*protected\_fifos*, but allows an application to avoid writes

to an attacker-controlled regular file, where the

application expected to create one.

*/proc/sys/fs/protected\_symlinks* (since Linux 3.6)

When the value in this file is 0, no restrictions are

placed on following symbolic links (i.e., this is the

historical behavior before Linux 3.6). When the value in

this file is 1, symbolic links are followed only in the

following circumstances:

\* the filesystem UID of the process following the link

matches the owner (UID) of the symbolic link (as

described in [credentials(7)](https://man7.org/linux/man-pages/man7/credentials.7.html), a process's filesystem UID

is normally the same as its effective UID);

\* the link is not in a sticky world-writable directory;

or

\* the symbolic link and its parent directory have the

same owner (UID)

A system call that fails to follow a symbolic link because

of the above restrictions returns the error **EACCES** in

[*errno*](https://man7.org/linux/man-pages/man3/errno.3.html).

The default value in this file is 0. Setting the value to

1 avoids a longstanding class of security issues based on

time-of-check, time-of-use races when accessing symbolic

links.

*/proc/sys/fs/suid\_dumpable* (since Linux 2.6.13)

The value in this file is assigned to a process's

"dumpable" flag in the circumstances described in

[prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html). In effect, the value in this file determines

whether core dump files are produced for set-user-ID or

otherwise protected/tainted binaries. The "dumpable"

setting also affects the ownership of files in a process's

*/proc/[pid]* directory, as described above.

Three different integer values can be specified:

*0 (default)*

This provides the traditional (pre-Linux 2.6.13)

behavior. A core dump will not be produced for a

process which has changed credentials (by calling

[seteuid(2)](https://man7.org/linux/man-pages/man2/seteuid.2.html), [setgid(2)](https://man7.org/linux/man-pages/man2/setgid.2.html), or similar, or by executing

a set-user-ID or set-group-ID program) or whose

binary does not have read permission enabled.

*1 ("debug")*

All processes dump core when possible. (Reasons

why a process might nevertheless not dump core are

described in [core(5)](https://man7.org/linux/man-pages/man5/core.5.html).) The core dump is owned by

the filesystem user ID of the dumping process and

no security is applied. This is intended for

system debugging situations only: this mode is

insecure because it allows unprivileged users to

examine the memory contents of privileged

processes.

*2 ("suidsafe")*

Any binary which normally would not be dumped (see

"0" above) is dumped readable by root only. This

allows the user to remove the core dump file but

not to read it. 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.

Additionally, since Linux 3.6,

*/proc/sys/kernel/core\_pattern* must either be an

absolute pathname or a pipe command, as detailed in

[core(5)](https://man7.org/linux/man-pages/man5/core.5.html). Warnings will be written to the kernel

log if *core\_pattern* does not follow these rules,

and no core dump will be produced.

For details of the effect of a process's "dumpable"

setting on ptrace access mode checking, see [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/sys/fs/super-max*

This file controls the maximum number of superblocks, and

thus the maximum number of mounted filesystems the kernel

can have. You need increase only *super-max* if you need to

mount more filesystems than the current value in *super-max*

allows you to.

*/proc/sys/fs/super-nr*

This file contains the number of filesystems currently

mounted.

*/proc/sys/kernel*

This directory contains files controlling a range of

kernel parameters, as described below.

*/proc/sys/kernel/acct*

This file contains three numbers: *highwater*, *lowwater*, and

*frequency*. If BSD-style process accounting is enabled,

these values control its behavior. If free space on

filesystem where the log lives goes below *lowwater*

percent, accounting suspends. If free space gets above

*highwater* percent, accounting resumes. *frequency*

determines how often the kernel checks the amount of free

space (value is in seconds). Default values are 4, 2, and

30. That is, suspend accounting if 2% or less space is

free; resume it if 4% or more space is free; consider

information about amount of free space valid for 30

seconds.

*/proc/sys/kernel/auto\_msgmni* (Linux 2.6.27 to 3.18)

From Linux 2.6.27 to 3.18, this file was used to control

recomputing of the value in */proc/sys/kernel/msgmni* upon

the addition or removal of memory or upon IPC namespace

creation/removal. Echoing "1" into this file enabled

*msgmni* automatic recomputing (and triggered a

recomputation of *msgmni* based on the current amount of

available memory and number of IPC namespaces). Echoing

"0" disabled automatic recomputing. (Automatic

recomputing was also disabled if a value was explicitly

assigned to */proc/sys/kernel/msgmni*.) The default value

in *auto\_msgmni* was 1.

Since Linux 3.19, the content of this file has no effect

(because *msgmni* defaults to near the maximum value

possible), and reads from this file always return the

value "0".

*/proc/sys/kernel/cap\_last\_cap* (since Linux 3.2)

See [capabilities(7)](https://man7.org/linux/man-pages/man7/capabilities.7.html).

*/proc/sys/kernel/cap-bound* (from Linux 2.2 to 2.6.24)

This file holds the value of the kernel *capability*

*bounding set* (expressed as a signed decimal number). This

set is ANDed against the capabilities permitted to a

process during [execve(2)](https://man7.org/linux/man-pages/man2/execve.2.html). Starting with Linux 2.6.25, the

system-wide capability bounding set disappeared, and was

replaced by a per-thread bounding set; see

[capabilities(7)](https://man7.org/linux/man-pages/man7/capabilities.7.html).

*/proc/sys/kernel/core\_pattern*

See [core(5)](https://man7.org/linux/man-pages/man5/core.5.html).

*/proc/sys/kernel/core\_pipe\_limit*

See [core(5)](https://man7.org/linux/man-pages/man5/core.5.html).

*/proc/sys/kernel/core\_uses\_pid*

See [core(5)](https://man7.org/linux/man-pages/man5/core.5.html).

*/proc/sys/kernel/ctrl-alt-del*

This file controls the handling of Ctrl-Alt-Del from the

keyboard. When the value in this file is 0, Ctrl-Alt-Del

is trapped and sent to the [init(1)](https://man7.org/linux/man-pages/man1/init.1.html) program to handle a

graceful restart. When the value is greater than zero,

Linux's reaction to a Vulcan Nerve Pinch (tm) will be an

immediate reboot, without even syncing its dirty buffers.

Note: when a program (like dosemu) has the keyboard in

"raw" mode, the ctrl-alt-del is intercepted by the program

before it ever reaches the kernel tty layer, and it's up

to the program to decide what to do with it.

*/proc/sys/kernel/dmesg\_restrict* (since Linux 2.6.37)

The value in this file determines who can see kernel

syslog contents. A value of 0 in this file imposes no

restrictions. If the value is 1, only privileged users

can read the kernel syslog. (See [syslog(2)](https://man7.org/linux/man-pages/man2/syslog.2.html) for more

details.) Since Linux 3.4, only users with the

**CAP\_SYS\_ADMIN** capability may change the value in this

file.

*/proc/sys/kernel/domainname* and */proc/sys/kernel/hostname*

can be used to set the NIS/YP domainname and the hostname

of your box in exactly the same way as the commands

[domainname(1)](https://man7.org/linux/man-pages/man1/domainname.1.html) and [hostname(1)](https://man7.org/linux/man-pages/man1/hostname.1.html), that is:

# **echo 'darkstar' > /proc/sys/kernel/hostname**

# **echo 'mydomain' > /proc/sys/kernel/domainname**

has the same effect as

# **hostname 'darkstar'**

# **domainname 'mydomain'**

Note, however, that the classic darkstar.frop.org has the

hostname "darkstar" and DNS (Internet Domain Name Server)

domainname "frop.org", not to be confused with the NIS

(Network Information Service) or YP (Yellow Pages)

domainname. These two domain names are in general

different. For a detailed discussion see the [hostname(1)](https://man7.org/linux/man-pages/man1/hostname.1.html)

man page.

*/proc/sys/kernel/hotplug*

This file contains the pathname for the hotplug policy

agent. The default value in this file is */sbin/hotplug*.

*/proc/sys/kernel/htab-reclaim* (before Linux 2.4.9.2)

(PowerPC only) If this file is set to a nonzero value, the

PowerPC htab (see kernel file

*Documentation/powerpc/ppc\_htab.txt*) is pruned each time

the system hits the idle loop.

*/proc/sys/kernel/keys/\**

This directory contains various files that define

parameters and limits for the key-management facility.

These files are described in [keyrings(7)](https://man7.org/linux/man-pages/man7/keyrings.7.html).

*/proc/sys/kernel/kptr\_restrict* (since Linux 2.6.38)

The value in this file determines whether kernel addresses

are exposed via */proc* files and other interfaces. A value

of 0 in this file imposes no restrictions. If the value

is 1, kernel pointers printed using the *%pK* format

specifier will be replaced with zeros unless the user has

the **CAP\_SYSLOG** capability. If the value is 2, kernel

pointers printed using the *%pK* format specifier will be

replaced with zeros regardless of the user's capabilities.

The initial default value for this file was 1, but the

default was changed to 0 in Linux 2.6.39. Since Linux

3.4, only users with the **CAP\_SYS\_ADMIN** capability can

change the value in this file.

*/proc/sys/kernel/l2cr*

(PowerPC only) This file contains a flag that controls the

L2 cache of G3 processor boards. If 0, the cache is

disabled. Enabled if nonzero.

*/proc/sys/kernel/modprobe*

This file contains the pathname for the kernel module

loader. The default value is */sbin/modprobe*. The file is

present only if the kernel is built with the

**CONFIG\_MODULES** (**CONFIG\_KMOD** in Linux 2.6.26 and earlier)

option enabled. It is described by the Linux kernel

source file *Documentation/kmod.txt* (present only in kernel

2.4 and earlier).

*/proc/sys/kernel/modules\_disabled* (since Linux 2.6.31)

A toggle value indicating if modules are allowed to be

loaded in an otherwise modular kernel. This toggle

defaults to off (0), but can be set true (1). Once true,

modules can be neither loaded nor unloaded, and the toggle

cannot be set back to false. The file is present only if

the kernel is built with the **CONFIG\_MODULES** option

enabled.

*/proc/sys/kernel/msgmax* (since Linux 2.2)

This file defines a system-wide limit specifying the

maximum number of bytes in a single message written on a

System V message queue.

*/proc/sys/kernel/msgmni* (since Linux 2.4)

This file defines the system-wide limit on the number of

message queue identifiers. See also

*/proc/sys/kernel/auto\_msgmni*.

*/proc/sys/kernel/msgmnb* (since Linux 2.2)

This file defines a system-wide parameter used to

initialize the *msg\_qbytes* setting for subsequently created

message queues. The *msg\_qbytes* setting specifies the

maximum number of bytes that may be written to the message

queue.

*/proc/sys/kernel/ngroups\_max* (since Linux 2.6.4)

This is a read-only file that displays the upper limit on

the number of a process's group memberships.

*/proc/sys/kernel/ns\_last\_pid* (since Linux 3.3)

See [pid\_namespaces(7)](https://man7.org/linux/man-pages/man7/pid_namespaces.7.html).

*/proc/sys/kernel/ostype* and */proc/sys/kernel/osrelease*

These files give substrings of */proc/version*.

*/proc/sys/kernel/overflowgid* and */proc/sys/kernel/overflowuid*

These files duplicate the files */proc/sys/fs/overflowgid*

and */proc/sys/fs/overflowuid*.

*/proc/sys/kernel/panic*

This file gives read/write access to the kernel variable

*panic\_timeout*. If this is zero, the kernel will loop on a

panic; if nonzero, it indicates that the kernel should

autoreboot after this number of seconds. When you use the

software watchdog device driver, the recommended setting

is 60.

*/proc/sys/kernel/panic\_on\_oops* (since Linux 2.5.68)

This file controls the kernel's behavior when an oops or

BUG is encountered. If this file contains 0, then the

system tries to continue operation. If it contains 1,

then the system delays a few seconds (to give klogd time

to record the oops output) and then panics. If the

*/proc/sys/kernel/panic* file is also nonzero, then the

machine will be rebooted.

*/proc/sys/kernel/pid\_max* (since Linux 2.5.34)

This file specifies the value at which PIDs wrap around

(i.e., the value in this file is one greater than the

maximum PID). PIDs greater than this value are not

allocated; thus, the value in this file also acts as a

system-wide limit on the total number of processes and

threads. The default value for this file, 32768, results

in the same range of PIDs as on earlier kernels. On

32-bit platforms, 32768 is the maximum value for *pid\_max*.

On 64-bit systems, *pid\_max* can be set to any value up to

2^22 (**PID\_MAX\_LIMIT**, approximately 4 million).

*/proc/sys/kernel/powersave-nap* (PowerPC only)

This file contains a flag. If set, Linux-PPC will use the

"nap" mode of powersaving, otherwise the "doze" mode will

be used.

*/proc/sys/kernel/printk*

See [syslog(2)](https://man7.org/linux/man-pages/man2/syslog.2.html).

*/proc/sys/kernel/pty* (since Linux 2.6.4)

This directory contains two files relating to the number

of UNIX 98 pseudoterminals (see [pts(4)](https://man7.org/linux/man-pages/man4/pts.4.html)) on the system.

*/proc/sys/kernel/pty/max*

This file defines the maximum number of pseudoterminals.

*/proc/sys/kernel/pty/nr*

This read-only file indicates how many pseudoterminals are

currently in use.

*/proc/sys/kernel/random*

This directory contains various parameters controlling the

operation of the file */dev/random*. See [random(4)](https://man7.org/linux/man-pages/man4/random.4.html) for

further information.

*/proc/sys/kernel/random/uuid* (since Linux 2.4)

Each read from this read-only file returns a randomly

generated 128-bit UUID, as a string in the standard UUID

format.

*/proc/sys/kernel/randomize\_va\_space* (since Linux 2.6.12)

Select the address space layout randomization (ASLR)

policy for the system (on architectures that support

ASLR). Three values are supported for this file:

0 Turn ASLR off. This is the default for architectures

that don't support ASLR, and when the kernel is booted

with the *norandmaps* parameter.

1 Make the addresses of [mmap(2)](https://man7.org/linux/man-pages/man2/mmap.2.html) allocations, the stack,

and the VDSO page randomized. Among other things, this

means that shared libraries will be loaded at

randomized addresses. The text segment of PIE-linked

binaries will also be loaded at a randomized address.

This value is the default if the kernel was configured

with **CONFIG\_COMPAT\_BRK**.

2 (Since Linux 2.6.25) Also support heap randomization.

This value is the default if the kernel was not

configured with **CONFIG\_COMPAT\_BRK**.

*/proc/sys/kernel/real-root-dev*

This file is documented in the Linux kernel source file

*Documentation/admin-guide/initrd.rst* (or

*Documentation/initrd.txt* before Linux 4.10).

*/proc/sys/kernel/reboot-cmd* (Sparc only)

This file seems to be a way to give an argument to the

SPARC ROM/Flash boot loader. Maybe to tell it what to do

after rebooting?

*/proc/sys/kernel/rtsig-max*

(Only in kernels up to and including 2.6.7; see

[setrlimit(2)](https://man7.org/linux/man-pages/man2/setrlimit.2.html)) This file can be used to tune the maximum

number of POSIX real-time (queued) signals that can be

outstanding in the system.

*/proc/sys/kernel/rtsig-nr*

(Only in kernels up to and including 2.6.7.) This file

shows the number of POSIX real-time signals currently

queued.

*/proc/[pid]/sched\_autogroup\_enabled* (since Linux 2.6.38)

See [sched(7)](https://man7.org/linux/man-pages/man7/sched.7.html).

***/proc/sys/kernel/sched\_child\_runs\_first* (since Linux 2.6.23)**

If this file contains the value zero, then, after a

[fork(2)](https://man7.org/linux/man-pages/man2/fork.2.html), the parent is first scheduled on the CPU. If the

file contains a nonzero value, then the child is scheduled

first on the CPU. (Of course, on a multiprocessor system,

the parent and the child might both immediately be

scheduled on a CPU.)

*/proc/sys/kernel/sched\_rr\_timeslice\_ms* (since Linux 3.9)

See [sched\_rr\_get\_interval(2)](https://man7.org/linux/man-pages/man2/sched_rr_get_interval.2.html).

***/proc/sys/kernel/sched\_rt\_period\_us* (since Linux 2.6.25)**

See [sched(7)](https://man7.org/linux/man-pages/man7/sched.7.html).

*/proc/sys/kernel/sched\_rt\_runtime\_us* (since Linux 2.6.25)

See [sched(7)](https://man7.org/linux/man-pages/man7/sched.7.html).

*/proc/sys/kernel/seccomp* (since Linux 4.14)

This directory provides additional seccomp information and

configuration. See [seccomp(2)](https://man7.org/linux/man-pages/man2/seccomp.2.html) for further details.

***/proc/sys/kernel/sem* (since Linux 2.4)**

This file contains 4 numbers defining limits for System V

IPC semaphores. These fields are, in order:

SEMMSL The maximum semaphores per semaphore set.

SEMMNS A system-wide limit on the number of semaphores in

all semaphore sets.

SEMOPM The maximum number of operations that may be

specified in a [semop(2)](https://man7.org/linux/man-pages/man2/semop.2.html) call.

SEMMNI A system-wide limit on the maximum number of

semaphore identifiers.

*/proc/sys/kernel/sg-big-buff*

This file shows the size of the generic SCSI device (sg)

buffer. You can't tune it just yet, but you could change

it at compile time by editing *include/scsi/sg.h* and

changing the value of **SG\_BIG\_BUFF**. However, there

shouldn't be any reason to change this value.

*/proc/sys/kernel/shm\_rmid\_forced* (since Linux 3.1)

If this file is set to 1, all System V shared memory

segments will be marked for destruction as soon as the

number of attached processes falls to zero; in other

words, it is no longer possible to create shared memory

segments that exist independently of any attached process.

The effect is as though a [shmctl(2)](https://man7.org/linux/man-pages/man2/shmctl.2.html) **IPC\_RMID** is performed

on all existing segments as well as all segments created

in the future (until this file is reset to 0). Note that

existing segments that are attached to no process will be

immediately destroyed when this file is set to 1. Setting

this option will also destroy segments that were created,

but never attached, upon termination of the process that

created the segment with [shmget(2)](https://man7.org/linux/man-pages/man2/shmget.2.html).

Setting this file to 1 provides a way of ensuring that all

System V shared memory segments are counted against the

resource usage and resource limits (see the description of

**RLIMIT\_AS** in [getrlimit(2)](https://man7.org/linux/man-pages/man2/getrlimit.2.html)) of at least one process.

Because setting this file to 1 produces behavior that is

nonstandard and could also break existing applications,

the default value in this file is 0. Set this file to 1

only if you have a good understanding of the semantics of

the applications using System V shared memory on your

system.

*/proc/sys/kernel/shmall* (since Linux 2.2)

This file contains the system-wide limit on the total

number of pages of System V shared memory.

*/proc/sys/kernel/shmmax* (since Linux 2.2)

This file can be used to query and set the run-time limit

on the maximum (System V IPC) shared memory segment size

that can be created. Shared memory segments up to 1 GB

are now supported in the kernel. This value defaults to

**SHMMAX**.

*/proc/sys/kernel/shmmni* (since Linux 2.4)

This file specifies the system-wide maximum number of

System V shared memory segments that can be created.

*/proc/sys/kernel/sysctl\_writes\_strict* (since Linux 3.16)

The value in this file determines how the file offset

affects the behavior of updating entries in files under

*/proc/sys*. The file has three possible values:

-1 This provides legacy handling, with no printk

warnings. Each [write(2)](https://man7.org/linux/man-pages/man2/write.2.html) must fully contain the value

to be written, and multiple writes on the same file

descriptor will overwrite the entire value, regardless

of the file position.

0 (default) This provides the same behavior as for -1,

but printk warnings are written for processes that

perform writes when the file offset is not 0.

1 Respect the file offset when writing strings into

*/proc/sys* files. Multiple writes will *append* to the

value buffer. Anything written beyond the maximum

length of the value buffer will be ignored. Writes to

numeric */proc/sys* entries must always be at file

offset 0 and the value must be fully contained in the

buffer provided to [write(2)](https://man7.org/linux/man-pages/man2/write.2.html).

*/proc/sys/kernel/sysrq*

This file controls the functions allowed to be invoked by

the SysRq key. By default, the file contains 1 meaning

that every possible SysRq request is allowed (in older

kernel versions, SysRq was disabled by default, and you

were required to specifically enable it at run-time, but

this is not the case any more). Possible values in this

file are:

0 Disable sysrq completely

1 Enable all functions of sysrq

> 1 Bit mask of allowed sysrq functions, as follows:

2 Enable control of console logging level

4 Enable control of keyboard (SAK, unraw)

8 Enable debugging dumps of processes etc.

16 Enable sync command

32 Enable remount read-only

64 Enable signaling of processes (term, kill, oom-

kill)

128 Allow reboot/poweroff

256 Allow nicing of all real-time tasks

This file is present only if the **CONFIG\_MAGIC\_SYSRQ** kernel

configuration option is enabled. For further details see

the Linux kernel source file

*Documentation/admin-guide/sysrq.rst* (or

*Documentation/sysrq.txt* before Linux 4.10).

*/proc/sys/kernel/version*

This file contains a string such as:

#5 Wed Feb 25 21:49:24 MET 1998

The "#5" means that this is the fifth kernel built from

this source base and the date following it indicates the

time the kernel was built.

*/proc/sys/kernel/threads-max* (since Linux 2.3.11)

This file specifies the system-wide limit on the number of

threads (tasks) that can be created on the system.

Since Linux 4.1, the value that can be written to

*threads-max* is bounded. The minimum value that can be

written is 20. The maximum value that can be written is

given by the constant **FUTEX\_TID\_MASK** (0x3fffffff). If a

value outside of this range is written to *threads-max*, the

error **EINVAL** occurs.

The value written is checked against the available RAM

pages. If the thread structures would occupy too much

(more than 1/8th) of the available RAM pages, *threads-max*

is reduced accordingly.

*/proc/sys/kernel/yama/ptrace\_scope* (since Linux 3.5)

See [ptrace(2)](https://man7.org/linux/man-pages/man2/ptrace.2.html).

*/proc/sys/kernel/zero-paged* (PowerPC only)

This file contains a flag. When enabled (nonzero), Linux-

PPC will pre-zero pages in the idle loop, possibly

speeding up get\_free\_pages.

*/proc/sys/net*

This directory contains networking stuff. Explanations

for some of the files under this directory can be found in

[tcp(7)](https://man7.org/linux/man-pages/man7/tcp.7.html) and [ip(7)](https://man7.org/linux/man-pages/man7/ip.7.html).

*/proc/sys/net/core/bpf\_jit\_enable*

See [bpf(2)](https://man7.org/linux/man-pages/man2/bpf.2.html).

*/proc/sys/net/core/somaxconn*

This file defines a ceiling value for the *backlog* argument

of [listen(2)](https://man7.org/linux/man-pages/man2/listen.2.html); see the [listen(2)](https://man7.org/linux/man-pages/man2/listen.2.html) manual page for details.

*/proc/sys/proc*

This directory may be empty.

*/proc/sys/sunrpc*

This directory supports Sun remote procedure call for

network filesystem (NFS). On some systems, it is not

present.

*/proc/sys/user* (since Linux 4.9)

See [namespaces(7)](https://man7.org/linux/man-pages/man7/namespaces.7.html).

*/proc/sys/vm*

This directory contains files for memory management

tuning, buffer, and cache management.

*/proc/sys/vm/admin\_reserve\_kbytes* (since Linux 3.10)

This file defines the amount of free memory (in KiB) on

the system that should be reserved for users with the

capability **CAP\_SYS\_ADMIN**.

The default value in this file is the minimum of [3% of

free pages, 8MiB] expressed as KiB. The default is

intended to provide enough for the superuser to log in and

kill a process, if necessary, under the default overcommit

'guess' mode (i.e., 0 in */proc/sys/vm/overcommit\_memory*).

Systems running in "overcommit never" mode (i.e., 2 in

*/proc/sys/vm/overcommit\_memory*) should increase the value

in this file to account for the full virtual memory size

of the programs used to recover (e.g., [login(1)](https://man7.org/linux/man-pages/man1/login.1.html) [ssh(1)](https://man7.org/linux/man-pages/man1/ssh.1.html),

and [top(1)](https://man7.org/linux/man-pages/man1/top.1.html)) Otherwise, the superuser may not be able to

log in to recover the system. For example, on x86-64 a

suitable value is 131072 (128MiB reserved).

Changing the value in this file takes effect whenever an

application requests memory.

*/proc/sys/vm/compact\_memory* (since Linux 2.6.35)

When 1 is written to this file, all zones are compacted

such that free memory is available in contiguous blocks

where possible. The effect of this action can be seen by

examining */proc/buddyinfo*.

Present only if the kernel was configured with

**CONFIG\_COMPACTION**.

*/proc/sys/vm/drop\_caches* (since Linux 2.6.16)

Writing to this file causes the kernel to drop clean

caches, dentries, and inodes from memory, causing that

memory to become free. This can be useful for memory

management testing and performing reproducible filesystem

benchmarks. Because writing to this file causes the

benefits of caching to be lost, it can degrade overall

system performance.

To free pagecache, use:

echo 1 > /proc/sys/vm/drop\_caches

To free dentries and inodes, use:

echo 2 > /proc/sys/vm/drop\_caches

To free pagecache, dentries, and inodes, use:

echo 3 > /proc/sys/vm/drop\_caches

Because writing to this file is a nondestructive operation

and dirty objects are not freeable, the user should run

[sync(1)](https://man7.org/linux/man-pages/man1/sync.1.html) first.

*/proc/sys/vm/sysctl\_hugetlb\_shm\_group* (since Linux 2.6.7)

This writable file contains a group ID that is allowed to

allocate memory using huge pages. If a process has a

filesystem group ID or any supplementary group ID that

matches this group ID, then it can make huge-page

allocations without holding the **CAP\_IPC\_LOCK** capability;

see [memfd\_create(2)](https://man7.org/linux/man-pages/man2/memfd_create.2.html), [mmap(2)](https://man7.org/linux/man-pages/man2/mmap.2.html), and [shmget(2)](https://man7.org/linux/man-pages/man2/shmget.2.html).

*/proc/sys/vm/legacy\_va\_layout* (since Linux 2.6.9)

If nonzero, this disables the new 32-bit memory-mapping

layout; the kernel will use the legacy (2.4) layout for

all processes.

*/proc/sys/vm/memory\_failure\_early\_kill* (since Linux 2.6.32)

Control how to kill processes when an uncorrected memory

error (typically a 2-bit error in a memory module) that

cannot be handled by the kernel is detected in the

background by hardware. In some cases (like the page

still having a valid copy on disk), the kernel will handle

the failure transparently without affecting any

applications. But if there is no other up-to-date copy of

the data, it will kill processes to prevent any data

corruptions from propagating.

The file has one of the following values:

1: Kill all processes that have the corrupted-and-not-

reloadable page mapped as soon as the corruption is

detected. Note that this is not supported for a few

types of pages, such as kernel internally allocated

data or the swap cache, but works for the majority of

user pages.

0: Unmap the corrupted page from all processes and kill a

process only if it tries to access the page.

The kill is performed using a **SIGBUS** signal with *si\_code*

set to **BUS\_MCEERR\_AO**. Processes can handle this if they

want to; see [sigaction(2)](https://man7.org/linux/man-pages/man2/sigaction.2.html) for more details.

This feature is active only on architectures/platforms

with advanced machine check handling and depends on the

hardware capabilities.

Applications can override the *memory\_failure\_early\_kill*

setting individually with the [prctl(2)](https://man7.org/linux/man-pages/man2/prctl.2.html) **PR\_MCE\_KILL**

operation.

Present only if the kernel was configured with

**CONFIG\_MEMORY\_FAILURE**.

*/proc/sys/vm/memory\_failure\_recovery* (since Linux 2.6.32)

Enable memory failure recovery (when supported by the

platform).

1: Attempt recovery.

0: Always panic on a memory failure.

Present only if the kernel was configured with

**CONFIG\_MEMORY\_FAILURE**.

*/proc/sys/vm/oom\_dump\_tasks* (since Linux 2.6.25)

Enables a system-wide task dump (excluding kernel threads)

to be produced when the kernel performs an OOM-killing.

The dump includes the following information for each task

(thread, process): thread ID, real user ID, thread group

ID (process ID), virtual memory size, resident set size,

the CPU that the task is scheduled on, oom\_adj score (see

the description of */proc/[pid]/oom\_adj*), and command name.

This is helpful to determine why the OOM-killer was

invoked and to identify the rogue task that caused it.

If this contains the value zero, this information is

suppressed. On very large systems with thousands of

tasks, it may not be feasible to dump the memory state

information for each one. Such systems should not be

forced to incur a performance penalty in OOM situations

when the information may not be desired.

If this is set to nonzero, this information is shown

whenever the OOM-killer actually kills a memory-hogging

task.

The default value is 0.

*/proc/sys/vm/oom\_kill\_allocating\_task* (since Linux 2.6.24)

This enables or disables killing the OOM-triggering task

in out-of-memory situations.

If this is set to zero, the OOM-killer will scan through

the entire tasklist and select a task based on heuristics

to kill. This normally selects a rogue memory-hogging

task that frees up a large amount of memory when killed.

If this is set to nonzero, the OOM-killer simply kills the

task that triggered the out-of-memory condition. This

avoids a possibly expensive tasklist scan.

If */proc/sys/vm/panic\_on\_oom* is nonzero, it takes

precedence over whatever value is used in

*/proc/sys/vm/oom\_kill\_allocating\_task*.

The default value is 0.

*/proc/sys/vm/overcommit\_kbytes* (since Linux 3.14)

This writable file provides an alternative to

*/proc/sys/vm/overcommit\_ratio* for controlling the

*CommitLimit* when */proc/sys/vm/overcommit\_memory* has the

value 2. It allows the amount of memory overcommitting to

be specified as an absolute value (in kB), rather than as

a percentage, as is done with *overcommit\_ratio*. This

allows for finer-grained control of *CommitLimit* on systems

with extremely large memory sizes.

Only one of *overcommit\_kbytes* or *overcommit\_ratio* can have

an effect: if *overcommit\_kbytes* has a nonzero value, then

it is used to calculate *CommitLimit*, otherwise

*overcommit\_ratio* is used. Writing a value to either of

these files causes the value in the other file to be set

to zero.

*/proc/sys/vm/overcommit\_memory*

This file contains the kernel virtual memory accounting

mode. Values are:

0: heuristic overcommit (this is the default)

1: always overcommit, never check

2: always check, never overcommit

In mode 0, calls of [mmap(2)](https://man7.org/linux/man-pages/man2/mmap.2.html) with **MAP\_NORESERVE** are not

checked, and the default check is very weak, leading to

the risk of getting a process "OOM-killed".

In mode 1, the kernel pretends there is always enough

memory, until memory actually runs out. One use case for

this mode is scientific computing applications that employ

large sparse arrays. In Linux kernel versions before

2.6.0, any nonzero value implies mode 1.

In mode 2 (available since Linux 2.6), the total virtual

address space that can be allocated (*CommitLimit* in

*/proc/meminfo*) is calculated as

CommitLimit = (total\_RAM - total\_huge\_TLB) \*

overcommit\_ratio / 100 + total\_swap

where:

\* *total\_RAM* is the total amount of RAM on the

system;

\* *total\_huge\_TLB* is the amount of memory set aside

for huge pages;

\* *overcommit\_ratio* is the value in

*/proc/sys/vm/overcommit\_ratio*; and

\* *total\_swap* is the amount of swap space.

For example, on a system with 16 GB of physical RAM, 16 GB

of swap, no space dedicated to huge pages, and an

*overcommit\_ratio* of 50, this formula yields a *CommitLimit*

of 24 GB.

Since Linux 3.14, if the value in

*/proc/sys/vm/overcommit\_kbytes* is nonzero, then

*CommitLimit* is instead calculated as:

CommitLimit = overcommit\_kbytes + total\_swap

See also the description of

*/proc/sys/vm/admin\_reserve\_kbytes* and

*/proc/sys/vm/user\_reserve\_kbytes*.

*/proc/sys/vm/overcommit\_ratio* (since Linux 2.6.0)

This writable file defines a percentage by which memory

can be overcommitted. The default value in the file is

50. See the description of

*/proc/sys/vm/overcommit\_memory*.

*/proc/sys/vm/panic\_on\_oom* (since Linux 2.6.18)

This enables or disables a kernel panic in an out-of-

memory situation.

If this file is set to the value 0, the kernel's OOM-

killer will kill some rogue process. Usually, the OOM-

killer is able to kill a rogue process and the system will

survive.

If this file is set to the value 1, then the kernel

normally panics when out-of-memory happens. However, if a

process limits allocations to certain nodes using memory

policies ([mbind(2)](https://man7.org/linux/man-pages/man2/mbind.2.html) **MPOL\_BIND**) or cpusets ([cpuset(7)](https://man7.org/linux/man-pages/man7/cpuset.7.html)) and

those nodes reach memory exhaustion status, one process

may be killed by the OOM-killer. No panic occurs in this

case: because other nodes' memory may be free, this means

the system as a whole may not have reached an out-of-

memory situation yet.

If this file is set to the value 2, the kernel always

panics when an out-of-memory condition occurs.

The default value is 0. 1 and 2 are for failover of

clustering. Select either according to your policy of

failover.

*/proc/sys/vm/swappiness*

The value in this file controls how aggressively the

kernel will swap memory pages. Higher values increase

aggressiveness, lower values decrease aggressiveness. The

default value is 60.

*/proc/sys/vm/user\_reserve\_kbytes* (since Linux 3.10)

Specifies an amount of memory (in KiB) to reserve for user

processes. This is intended to prevent a user from

starting a single memory hogging process, such that they

cannot recover (kill the hog). The value in this file has

an effect only when */proc/sys/vm/overcommit\_memory* is set

to 2 ("overcommit never" mode). In this case, the system

reserves an amount of memory that is the minimum of [3% of

current process size, *user\_reserve\_kbytes*].

The default value in this file is the minimum of [3% of

free pages, 128MiB] expressed as KiB.

If the value in this file is set to zero, then a user will

be allowed to allocate all free memory with a single

process (minus the amount reserved by

*/proc/sys/vm/admin\_reserve\_kbytes*). Any subsequent

attempts to execute a command will result in "fork: Cannot

allocate memory".

Changing the value in this file takes effect whenever an

application requests memory.

*/proc/sys/vm/unprivileged\_userfaultfd* (since Linux 5.2)

This (writable) file exposes a flag that controls whether

unprivileged processes are allowed to employ

[userfaultfd(2)](https://man7.org/linux/man-pages/man2/userfaultfd.2.html). If this file has the value 1, then

unprivileged processes may use [userfaultfd(2)](https://man7.org/linux/man-pages/man2/userfaultfd.2.html). If this

file has the value 0, then only processes that have the

**CAP\_SYS\_PTRACE** capability may employ [userfaultfd(2)](https://man7.org/linux/man-pages/man2/userfaultfd.2.html). The

default value in this file is 1.

*/proc/sysrq-trigger* (since Linux 2.4.21)

Writing a character to this file triggers the same SysRq

function as typing ALT-SysRq-<character> (see the

description of */proc/sys/kernel/sysrq*). This file is

normally writable only by *root*. For further details see

the Linux kernel source file

*Documentation/admin-guide/sysrq.rst* (or

*Documentation/sysrq.txt* before Linux 4.10).

*/proc/sysvipc*

Subdirectory containing the pseudo-files *msg*, *sem* and *shm*.

These files list the System V Interprocess Communication

(IPC) objects (respectively: message queues, semaphores,

and shared memory) that currently exist on the system,

providing similar information to that available via

[ipcs(1)](https://man7.org/linux/man-pages/man1/ipcs.1.html). These files have headers and are formatted (one

IPC object per line) for easy understanding. [sysvipc(7)](https://man7.org/linux/man-pages/man7/sysvipc.7.html)

provides further background on the information shown by

these files.

*/proc/thread-self* (since Linux 3.17)

This directory refers to the thread accessing the */proc*

filesystem, and is identical to the */proc/self/task/[tid]*

directory named by the process thread ID (*[tid]*) of the

same thread.

*/proc/timer\_list* (since Linux 2.6.21)

This read-only file exposes a list of all currently

pending (high-resolution) timers, all clock-event sources,

and their parameters in a human-readable form.

*/proc/timer\_stats* (from Linux 2.6.21 until Linux 4.10)

This is a debugging facility to make timer (ab)use in a

Linux system visible to kernel and user-space developers.

It can be used by kernel and user-space developers to

verify that their code does not make undue use of timers.

The goal is to avoid unnecessary wakeups, thereby

optimizing power consumption.

If enabled in the kernel (**CONFIG\_TIMER\_STATS**), but not

used, it has almost zero run-time overhead and a

relatively small data-structure overhead. Even if

collection is enabled at run time, overhead is low: all

the locking is per-CPU and lookup is hashed.

The */proc/timer\_stats* file is used both to control

sampling facility and to read out the sampled information.

The *timer\_stats* functionality is inactive on bootup. A

sampling period can be started using the following

command:

# echo 1 > /proc/timer\_stats

The following command stops a sampling period:

# echo 0 > /proc/timer\_stats

The statistics can be retrieved by:

$ cat /proc/timer\_stats

While sampling is enabled, each readout from

*/proc/timer\_stats* will see newly updated statistics. Once

sampling is disabled, the sampled information is kept

until a new sample period is started. This allows

multiple readouts.

Sample output from */proc/timer\_stats*:

$ **cat /proc/timer\_stats**

Timer Stats Version: v0.3

Sample period: 1.764 s

Collection: active

255, 0 swapper/3 hrtimer\_start\_range\_ns (tick\_sched\_timer)

71, 0 swapper/1 hrtimer\_start\_range\_ns (tick\_sched\_timer)

58, 0 swapper/0 hrtimer\_start\_range\_ns (tick\_sched\_timer)

4, 1694 gnome-shell mod\_delayed\_work\_on (delayed\_work\_timer\_fn)

17, 7 rcu\_sched rcu\_gp\_kthread (process\_timeout)

...

1, 4911 kworker/u16:0 mod\_delayed\_work\_on (delayed\_work\_timer\_fn)

1D, 2522 kworker/0:0 queue\_delayed\_work\_on (delayed\_work\_timer\_fn)

1029 total events, 583.333 events/sec

The output columns are:

\* a count of the number of events, optionally (since

Linux 2.6.23) followed by the letter 'D' if this is a

deferrable timer;

\* the PID of the process that initialized the timer;

\* the name of the process that initialized the timer;

\* the function where the timer was initialized; and

\* (in parentheses) the callback function that is

associated with the timer.

During the Linux 4.11 development cycle, this file was

removed because of security concerns, as it exposes

information across namespaces. Furthermore, it is

possible to obtain the same information via in-kernel

tracing facilities such as ftrace.

*/proc/tty*

Subdirectory containing the pseudo-files and

subdirectories for tty drivers and line disciplines.

*/proc/uptime*

This file contains two numbers (values in seconds): the

uptime of the system (including time spent in suspend) and

the amount of time spent in the idle process.

*/proc/version*

This string identifies the kernel version that is

currently running. It includes the contents of

*/proc/sys/kernel/ostype*, */proc/sys/kernel/osrelease*, and

*/proc/sys/kernel/version*. For example:

Linux version 1.0.9 (quinlan@phaze) #1 Sat May 14 01:51:54 EDT 1994

*/proc/vmstat* (since Linux 2.6.0)

This file displays various virtual memory statistics.

Each line of this file contains a single name-value pair,

delimited by white space. Some lines are present only if

the kernel was configured with suitable options. (In some

cases, the options required for particular files have

changed across kernel versions, so they are not listed

here. Details can be found by consulting the kernel

source code.) The following fields may be present:

*nr\_free\_pages* (since Linux 2.6.31)

*nr\_alloc\_batch* (since Linux 3.12)

*nr\_inactive\_anon* (since Linux 2.6.28)

*nr\_active\_anon* (since Linux 2.6.28)

*nr\_inactive\_file* (since Linux 2.6.28)

*nr\_active\_file* (since Linux 2.6.28)

*nr\_unevictable* (since Linux 2.6.28)

*nr\_mlock* (since Linux 2.6.28)

*nr\_anon\_pages* (since Linux 2.6.18)

*nr\_mapped* (since Linux 2.6.0)

*nr\_file\_pages* (since Linux 2.6.18)

*nr\_dirty* (since Linux 2.6.0)

*nr\_writeback* (since Linux 2.6.0)

*nr\_slab\_reclaimable* (since Linux 2.6.19)

*nr\_slab\_unreclaimable* (since Linux 2.6.19)

*nr\_page\_table\_pages* (since Linux 2.6.0)

*nr\_kernel\_stack* (since Linux 2.6.32)

Amount of memory allocated to kernel stacks.

*nr\_unstable* (since Linux 2.6.0)

*nr\_bounce* (since Linux 2.6.12)

*nr\_vmscan\_write* (since Linux 2.6.19)

*nr\_vmscan\_immediate\_reclaim* (since Linux 3.2)

*nr\_writeback\_temp* (since Linux 2.6.26)

*nr\_isolated\_anon* (since Linux 2.6.32)

*nr\_isolated\_file* (since Linux 2.6.32)

*nr\_shmem* (since Linux 2.6.32)

Pages used by shmem and [tmpfs(5)](https://man7.org/linux/man-pages/man5/tmpfs.5.html).

*nr\_dirtied* (since Linux 2.6.37)

*nr\_written* (since Linux 2.6.37)

*nr\_pages\_scanned* (since Linux 3.17)

*numa\_hit* (since Linux 2.6.18)

*numa\_miss* (since Linux 2.6.18)

*numa\_foreign* (since Linux 2.6.18)

*numa\_interleave* (since Linux 2.6.18)

*numa\_local* (since Linux 2.6.18)

*numa\_other* (since Linux 2.6.18)

*workingset\_refault* (since Linux 3.15)

*workingset\_activate* (since Linux 3.15)

*workingset\_nodereclaim* (since Linux 3.15)

*nr\_anon\_transparent\_hugepages* (since Linux 2.6.38)

*nr\_free\_cma* (since Linux 3.7)

Number of free CMA (Contiguous Memory Allocator)

pages.

*nr\_dirty\_threshold* (since Linux 2.6.37)

*nr\_dirty\_background\_threshold* (since Linux 2.6.37)

*pgpgin* (since Linux 2.6.0)

*pgpgout* (since Linux 2.6.0)

*pswpin* (since Linux 2.6.0)

*pswpout* (since Linux 2.6.0)

*pgalloc\_dma* (since Linux 2.6.5)

*pgalloc\_dma32* (since Linux 2.6.16)

*pgalloc\_normal* (since Linux 2.6.5)

*pgalloc\_high* (since Linux 2.6.5)

*pgalloc\_movable* (since Linux 2.6.23)

*pgfree* (since Linux 2.6.0)

*pgactivate* (since Linux 2.6.0)

*pgdeactivate* (since Linux 2.6.0)

*pgfault* (since Linux 2.6.0)

*pgmajfault* (since Linux 2.6.0)

*pgrefill\_dma* (since Linux 2.6.5)

*pgrefill\_dma32* (since Linux 2.6.16)

*pgrefill\_normal* (since Linux 2.6.5)

*pgrefill\_high* (since Linux 2.6.5)

*pgrefill\_movable* (since Linux 2.6.23)

*pgsteal\_kswapd\_dma* (since Linux 3.4)

*pgsteal\_kswapd\_dma32* (since Linux 3.4)

*pgsteal\_kswapd\_normal* (since Linux 3.4)

*pgsteal\_kswapd\_high* (since Linux 3.4)

*pgsteal\_kswapd\_movable* (since Linux 3.4)

*pgsteal\_direct\_dma*

*pgsteal\_direct\_dma32* (since Linux 3.4)

*pgsteal\_direct\_normal* (since Linux 3.4)

*pgsteal\_direct\_high* (since Linux 3.4)

*pgsteal\_direct\_movable* (since Linux 2.6.23)

*pgscan\_kswapd\_dma*

*pgscan\_kswapd\_dma32* (since Linux 2.6.16)

*pgscan\_kswapd\_normal* (since Linux 2.6.5)

*pgscan\_kswapd\_high*

*pgscan\_kswapd\_movable* (since Linux 2.6.23)

*pgscan\_direct\_dma*

*pgscan\_direct\_dma32* (since Linux 2.6.16)

*pgscan\_direct\_normal*

*pgscan\_direct\_high*

*pgscan\_direct\_movable* (since Linux 2.6.23)

*pgscan\_direct\_throttle* (since Linux 3.6)

*zone\_reclaim\_failed* (since linux 2.6.31)

*pginodesteal* (since linux 2.6.0)

*slabs\_scanned* (since linux 2.6.5)

*kswapd\_inodesteal* (since linux 2.6.0)

*kswapd\_low\_wmark\_hit\_quickly* (since 2.6.33)

*kswapd\_high\_wmark\_hit\_quickly* (since 2.6.33)

*pageoutrun* (since Linux 2.6.0)

*allocstall* (since Linux 2.6.0)

*pgrotated* (since Linux 2.6.0)

*drop\_pagecache* (since Linux 3.15)

*drop\_slab* (since Linux 3.15)

*numa\_pte\_updates* (since Linux 3.8)

*numa\_huge\_pte\_updates* (since Linux 3.13)

*numa\_hint\_faults* (since Linux 3.8)

*numa\_hint\_faults\_local* (since Linux 3.8)

*numa\_pages\_migrated* (since Linux 3.8)

*pgmigrate\_success* (since Linux 3.8)

*pgmigrate\_fail* (since Linux 3.8)

*compact\_migrate\_scanned* (since Linux 3.8)

*compact\_free\_scanned* (since Linux 3.8)

*compact\_isolated* (since Linux 3.8)

*compact\_stall* (since Linux 2.6.35)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*compact\_fail* (since Linux 2.6.35)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*compact\_success* (since Linux 2.6.35)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*htlb\_buddy\_alloc\_success* (since Linux 2.6.26)

*htlb\_buddy\_alloc\_fail* (since Linux 2.6.26)

*unevictable\_pgs\_culled* (since Linux 2.6.28)

*unevictable\_pgs\_scanned* (since Linux 2.6.28)

*unevictable\_pgs\_rescued* (since Linux 2.6.28)

*unevictable\_pgs\_mlocked* (since Linux 2.6.28)

*unevictable\_pgs\_munlocked* (since Linux 2.6.28)

*unevictable\_pgs\_cleared* (since Linux 2.6.28)

*unevictable\_pgs\_stranded* (since Linux 2.6.28)

*thp\_fault\_alloc* (since Linux 2.6.39)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*thp\_fault\_fallback* (since Linux 2.6.39)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*thp\_collapse\_alloc* (since Linux 2.6.39)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*thp\_collapse\_alloc\_failed* (since Linux 2.6.39)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*thp\_split* (since Linux 2.6.39)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*thp\_zero\_page\_alloc* (since Linux 3.8)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*thp\_zero\_page\_alloc\_failed* (since Linux 3.8)

See the kernel source file

*Documentation/admin-guide/mm/transhuge.rst*.

*balloon\_inflate* (since Linux 3.18)

*balloon\_deflate* (since Linux 3.18)

*balloon\_migrate* (since Linux 3.18)

*nr\_tlb\_remote\_flush* (since Linux 3.12)

*nr\_tlb\_remote\_flush\_received* (since Linux 3.12)

*nr\_tlb\_local\_flush\_all* (since Linux 3.12)

*nr\_tlb\_local\_flush\_one* (since Linux 3.12)

*vmacache\_find\_calls* (since Linux 3.16)

*vmacache\_find\_hits* (since Linux 3.16)

*vmacache\_full\_flushes* (since Linux 3.19)

*/proc/zoneinfo* (since Linux 2.6.13)

This file displays information about memory zones. This

is useful for analyzing virtual memory behavior.

**NOTES**[**top**](https://man7.org/linux/man-pages/man5/proc.5.html#top_of_page)

Many files contain strings (e.g., the environment and command

line) that are in the internal format, with subfields terminated

by null bytes ('\0'). When inspecting such files, you may find

that the results are more readable if you use a command of the

following form to display them:

$ **cat** *file* **| tr '\000' '\n'**

This manual page is incomplete, possibly inaccurate, and is the

kind of thing that needs to be updated very often.

The Linux kernel source files:

*Documentation/filesystems/proc.txt*, *Documentation/sysctl/fs.txt*,

*Documentation/sysctl/kernel.txt*, *Documentation/sysctl/net.txt*,

and *Documentation/sysctl/vm.txt*.

**COLOPHON**[**top**](https://man7.org/linux/man-pages/man5/proc.5.html#top_of_page)

This page is part of release 5.13 of the Linux *man-pages* project.

A description of the project, information about reporting bugs,

and the latest version of this page, can be found at

<https://www.kernel.org/doc/man-pages/>.

**Linux 2021-08-27 PROC(5)**

[Copyright and license for this manual page](https://man7.org/linux/man-pages/man5/proc.5.license.html)

|  |  |  |
| --- | --- | --- |
| HTML rendering created 2021-08-27 by [Michael Kerrisk](https://man7.org/mtk/index.html), author of [*The Linux Programming Interface*](https://man7.org/tlpi/), maintainer of the [Linux *man-pages* project](https://www.kernel.org/doc/man-pages/).  For details of in-depth **Linux/UNIX system programming training courses** that I teach, look [here](https://man7.org/training/).  Hosting by [jambit GmbH](https://www.jambit.com/index_en.html). |  | [Cover of TLPI](https://man7.org/tlpi/) |