NAME

fanotify_init – create and initialize fanotify group

SYNOPSIS

#include <fcntl.h>
#include <sys/fanotify.h>

int fanotify_init(unsigned int *flags*, **unsigned int** *event_f_flags*);

DESCRIPTION

For an overview of the fanotify API, see **fanotify**(7).

fanotify_init() initializes a new fanotify group and returns a file descriptor for the event queue associated with the group.

The file descriptor is used in calls to **fanotify_mark**(2) to specify the files, directories, mounts or filesystems for which fanotify events shall be created. These events are received by reading from the file descriptor. Some events are only informative, indicating that a file has been accessed. Other events can be used to determine whether another application is permitted to access a file or directory. Permission to access filesystem objects is granted by writing to the file descriptor.

Multiple programs may be using the fanotify interface at the same time to monitor the same files.

In the current implementation, the number of fanotify groups per user is limited to 128. This limit cannot be overridden.

Calling **fanotify_init**() requires the **CAP_SYS_ADMIN** capability. This constraint might be relaxed in future versions of the API. Therefore, certain additional capability checks have been implemented as indicated below.

The *flags* argument contains a multi-bit field defining the notification class of the listening application and further single bit fields specifying the behavior of the file descriptor.

If multiple listeners for permission events exist, the notification class is used to establish the sequence in which the listeners receive the events.

Only one of the following notification classes may be specified in *flags*:

FAN_CLASS_PRE_CONTENT
This value allows the receipt of events notifying that a file has been accessed and events for permission decisions if a file may be accessed. It is intended for event listeners that need to access files before they contain their final data. This notification class might be used by hierarchical storage managers, for example.

FAN_CLASS_CONTENT
This value allows the receipt of events notifying that a file has been accessed and events for permission decisions if a file may be accessed. It is intended for event listeners that need to access files when they already contain their final content. This notification class might be used by malware detection programs, for example.

FAN_REPORT_FID (since Linux 5.1)
This value allows the receipt of events which contain additional information about the underlying filesystem object correlated to an event. An additional structure encapsulates the information about the object and is included alongside the generic event metadata structure. The file descriptor that is used to represent the object correlated to an event is instead substituted with a file handle. It is intended for applications that may find the use of a file handle to identify an object more suitable than a file descriptor. Additionally, it may be used for applications that are interested in directory entry events, such as **FAN_CREATE**, **FAN_ATTRIB**, **FAN_MOVE**, and **FAN_DELETE** for example. Note that the use of directory modification events are not supported when monitoring a mount point. The use of **FAN_CLASS_CONTENT** or **FAN_CLASS_PRE_CONTENT** is not permitted with this flag and will result in the error **EINVAL**. See **fanotify**(7) for additional information.

FAN_CLASS_NOTIF
This is the default value. It does not need to be specified. This value only allows the receipt of events notifying that a file has been accessed. Permission decisions before the file is accessed are not possible.

Listeners with different notification classes will receive events in the order **FAN_CLASS_PRE_CONTENT**, **FAN_CLASS_CONTENT**, **FAN_CLASS_NOTIF**. The order of notification for listeners in the same notification class is undefined.

The following bits can additionally be set in *flags*:

FAN_CLOEXEC
Set the close-on-exec flag (**FD_CLOEXEC**) on the new file descriptor. See the description of the **O_CLOEXEC** flag in **open**(2).

FAN_NONBLOCK
Enable the nonblocking flag (**O_NONBLOCK**) for the file descriptor. Reading from the file descriptor will not block. Instead, if no data is available, **read**(2) fails with the error **EAGAIN**.

FAN_UNLIMITED_QUEUE
Remove the limit of 16384 events for the event queue. Use of this flag requires the **CAP_SYS_ADMIN** capability.

FAN_UNLIMITED_MARKS
Remove the limit of 8192 marks. Use of this flag requires the **CAP_SYS_ADMIN** capability.

FAN_REPORT_TID (since Linux 4.20)
Report thread ID (TID) instead of process ID (PID) in the *pid* field of the *struct fanotify_event_metadata* supplied to **read**(2) (see **fanotify**(7)).

The *event_f_flags* argument defines the file status flags that will be set on the open file descriptions that are created for fanotify events. For details of these flags, see the description of the *flags* values in **open**(2). *event_f_flags* includes a multi-bit field for the access mode. This field can take the following values:

O_RDONLY
This value allows only read access.

O_WRONLY
This value allows only write access.

O_RDWR
This value allows read and write access.

Additional bits can be set in *event_f_flags*. The most useful values are:

O_LARGEFILE
Enable support for files exceeding 2 GB. Failing to set this flag will result in an **EOVERFLOW** error when trying to open a large file which is monitored by an fanotify group on a 32-bit system.

O_CLOEXEC (since Linux 3.18)
Enable the close-on-exec flag for the file descriptor. See the description of the **O_CLOEXEC** flag in **open**(2) for reasons why this may be useful.

The following are also allowable: **O_APPEND**, **O_DSYNC**, **O_NOATIME**, **O_NONBLOCK**, and **O_SYNC**. Specifying any other flag in *event_f_flags* yields the error **EINVAL** (but see BUGS).

RETURN VALUE

On success, **fanotify_init**() returns a new file descriptor. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EINVAL
An invalid value was passed in *flags* or *event_f_flags*. **FAN_ALL_INIT_FLAGS** (deprecated since Linux kernel version 4.20) defines all allowable bits for *flags*.

EMFILE
The number of fanotify groups for this user exceeds 128.

EMFILE
The per-process limit on the number of open file descriptors has been reached.

ENOMEM
The allocation of memory for the notification group failed.

ENOSYS
This kernel does not implement **fanotify_init**(). The fanotify API is available only if the kernel was configured with **CONFIG_FANOTIFY**.

EPERM
The operation is not permitted because the caller lacks the **CAP_SYS_ADMIN** capability.

VERSIONS

fanotify_init() was introduced in version 2.6.36 of the Linux kernel and enabled in version 2.6.37.

CONFORMING TO

This system call is Linux-specific.

BUGS

The following bug was present in Linux kernels before version 3.18:

* The **O_CLOEXEC** is ignored when passed in *event_f_flags*.

The following bug was present in Linux kernels before version 3.14:

* The *event_f_flags* argument is not checked for invalid flags. Flags that are intended only for internal use, such as **FMODE_EXEC**, can be set, and will consequently be set for the file descriptors returned when reading from the fanotify file descriptor.

SEE ALSO

fanotify_mark(2), **fanotify**(7)

COLOPHON

This page is part of release 5.05 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/.