

**NAME**

**libinotify**, **inotify\_init**, **inotify\_init1**, **inotify\_add\_watch**, **inotify\_rm\_watch**, **libinotify\_set\_param**, **inotify\_event**, - monitor file system events

**SYNOPSIS**

```
#include <sys/inotify.h>
```

*int*

```
inotify_init();
```

*int*

```
inotify_init1(int flags);
```

*int*

```
inotify_add_watch(int fd, const char *pathname, uint32_t mask);
```

*int*

```
inotify_rm_watch(int fd, int wd);
```

*int*

```
libinotify_set_param(int fd, int param, intptr_t value);
```

**DESCRIPTION**

The **inotify\_init**() and **inotify\_init1**() functions create an inotify instance and returns a file descriptor referring to the instance. **inotify\_init1**() function is similar to **inotify\_init**() except that it takes additional flags parameter whose values can be:

**IN\_NONBLOCK**     Set I\_NONBLOCK file status flag on the inotify file descriptor.

**IN\_CLOSEEXEC**    Set FD\_CLOEXEC flag on the new file descriptor. See O\_CLOEXEC flag in `open(2)`

The function returns the file descriptor to the inotify handle if successful otherwise return -1. Possible errorno values are -

**EINVAL**           Invalid flag value passed.

**EMFILE**           System wide limit of inotify instances reached.

**ENFILE**           System limit on total number of fd's reached.

ENOMEM           Insufficient kernel memory.

**inotify\_add\_watch()** function adds news watch to the inotify instance. List of possible masks are described below. If the watch for given filename already exists, it is updated with the new mask value passed. The function returns an integer called watch descriptor if successful otherwise -1.

Possible values for errno are -

EACCES           Permission for read access is denied for given file.

EBADF            Invalid file descriptor.

EFAULT           Pathname points outside process's allocated address space.

EINVAL           Invalid event mask passed.

ENOENT           A component of path that must exist does not exist.

ENOMEM           Insufficient kernel memory available.

ENOSPC           User limit on total number of inotify watches has crossed or kernel failed to allocate a needed resource.

**inotify\_rm\_watch()** function removes watch wd from the instance described by file descriptor fd. The function returns zero on success and -1 on error. Possible errno values are -

EBADF            Invalid file descriptor fd.

EINVAL           Invalid watch descriptor wd.

**libinotify\_set\_param()** Libinotify specific. Replacement for Linux procfs interface. Set inotify parameter for the instance described by file descriptor fd. fd value of -1 is used for setting of global parameters. Possible param values are -

IN\_SOCKBUFSIZE   Size of communication socket buffer in bytes. Should match read(2) buffer size for libinotify event consumers. Lower values can cause partial event reads. Bigger values is just a wasting of memory. Default value is arbitrary, has been acquired from code sample in linux inotify(7) man page and seems to be very common among the inotify clients. Default value 4096 (exported as IN\_DEF\_SOCKBUFSIZE)

**IN\_MAX\_QUEUED\_EVENTS**

Upper limit on the queue length per inotify handle. linux's `/proc/sys/fs/inotify/max_queued_events` counterpart. Default value 16384 (exported as `IN_DEF_MAX_QUEUED_EVENTS`)

**IN\_MAX\_USER\_INSTANCES**

Global upper limit on the number of inotify instances that can be created. linux's `/proc/sys/fs/inotify/max_user_instances` counterpart. Default value 2147483646 (exported as `IN_DEF_MAX_USER_INSTANCES`)

**inotify\_event structure**

```
struct inotify_event {
    int    wd; /* Watch descriptor */
    uint32_t mask; /* Mask of events */
    uint32_t cookie; /* Unique integer associating related events */
    uint32_t len; /* Size of name field */
    char    name[]; /* Optional null-terminated name */
};
```

**inotify events -**

Following are the masks supported by libinotify implementation.

<code>IN_OPEN</code>	File was opened.
<code>IN_ACCESS</code>	File was accessed (read).
<code>IN_ATTRIB</code>	Metadata changed.
<code>IN_CREATE</code>	File/directory was created in watched directory.
<code>IN_CLOSE_WRITE</code>	File opened for writing was closed.
<code>IN_CLOSE_NOWRITE</code>	File not opened for writing was closed.
<code>IN_DELETE</code>	File/directory in watched directory was deleted.
<code>IN_DELETE_SELF</code>	Watched file/directory was deleted.
<code>IN_MODIFY</code>	File/Directory was modified.
<code>IN_MOVE_SELF</code>	Watched file/directory was moved.
<code>IN_MOVED_FROM</code>	A file in watched directory was moved out.
<code>IN_MOVED_TO</code>	A file was moved into watched directory.
<code>IN_ALL_EVENTS</code>	Bit mask of all the above events.
<code>IN_MOVE</code>	Equal to <code>IN_MOVED_FROM IN_MOVED_TO</code>
<code>IN_CLOSE</code>	Equal to <code>IN_CLOSE_WRITE IN_CLOSE_NOWRITE</code>

`IN_DELETE_SELF` and `IN_MOVE_SELF` can occur only for watched file/directory. Other events can be marked for a file/directory in a watched directory. In that case the name of the file for which event is generated can be read by 'name' field in `inotify_event` structure.

Following are additional bits that can be set in mask when calling **inotify\_add\_watch()** -

**IN\_DONT\_FOLLOW**

Don't dereference path name if its symlink.

**IN\_EXCL\_UNLINK** Do not generate events for unlinked childrens. (Currently not supported).

**IN\_MASK\_ADD** Add event mask for watch for given pathname.

**IN\_ONESHOT** Remove watch after retrieving one event.

**IN\_ONLYDIR** Only watch the pathname if it is a directory.

Following bits may be set by mask field returned by read(3)

**IN\_IGNORED** Watch for removed (explicitely, revoked or unmounted).

**IN\_ISDIR** Subject of this event is a directory.

**IN\_Q\_OVERFLOW** Event queue has overflowed.

**IN\_UNMOUNT** File system containing watched file/directory was unmounted.

## SEE ALSO

read(3)

## HISTORY

inotify first appeared in Linux 2.6.13